AFRL-IF-WP-TR-2006-1578

ENLIGHTENED MULTISCALE SIMULATION OF BIOCHEMICAL NETWORKS

Core Theory, Validating Experiments, and Implementation in Open Software

John C. Doyle
Michael Hucka

California Institute of Technology
1200 E. California Blvd.
Pasadena, CA 91125

OCTOBER 2006

Final Report for 01 September 2001 – 31 July 2006

Approved for public release; distribution is unlimited.

STINFO COPY

INFORMATION DIRECTORATE
AIR FORCE RESEARCH LABORATORY
AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OH 45433-7334

NOTICE AND SIGNATURE PAGE

AFRL-IF-WP-TR-2006-1578 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

*//Signature//

JAMES B. MONCRIEF, Proj Eng
Embedded Information Systems Branch
Advanced Computing Division
Information Directorate

//Signature//

KENNETH LITTLEJOHN, Actg Chief
Embedded Information Systems Branch
Advanced Computing Division
Information Directorate

//Signature//

JAMES S. WILLIAMSON, Actg Chief
Wright Site
Information Directorate

*Disseminated copies will show "//signature//" stamped or typed above the signature blocks.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
October 2006	Final	09/01/2001 – 07/31/2006

4. TITLE AND SUBTITLE
ENLIGHTENED MULTISCALE SIMULATION OF BIOCHEMICAL NETWORKS
Core Theory, Validating Experiments, and Implementation in Open Software

5a. CONTRACT NUMBER
F30602-01-2-0558

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
61101E

6. AUTHOR(S)
John C. Doyle
Michael Hucka

5d. PROJECT NUMBER
BIOC

5e. TASK NUMBER
M3

5f. WORK UNIT NUMBER
05

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
California Institute of Technology
1200 E. California Blvd.
Pasadena, CA 91125

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Information Directorate
Air Force Research Laboratory
Air Force Materiel Command
Wright-Patterson AFB, OH 45433-7334

Defense Advanced Research Projects Agency/Information Processing Technology Office (DARPA/IPTO)
3701 Fairfax Drive
Arlington, VA 22203

10. SPONSORING/MONITORING AGENCY ACRONYM(S)
AFRL-IF-WP

11. SPONSORING/MONITORING AGENCY REPORT NUMBER(S)
AFRL-IF-WP-TR-2006-1578

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
Report contains color. PAO Case Number: AFRL/WS 07-0326, 21 Feb 2007.

14. ABSTRACT
The objective of the research is to develop mathematical and software infrastructure in support of post-genomics research in systems biology. One objective articulated in this effort centers on a deeper understanding of the organizational principles of biological networks. A distinguishing theme of this work is its focus on scalable methods of robustness and model validation and invalidation with data, as opposed to relying purely on simulation. The Systems Biology Markup Language (SBML) project is a machine-readable exchange language for computational models of biochemical networks. LibSBML, an embedded software library for SBML, was developed, providing an application programming interface for working with SBML. The LibSBML library provides an interface for working with SBML in a number of programming languages: C, C++, Java, Perl, MATLAB, Lisp, and Python. It is free, open-source, and portable to Linux, Windows, MacOS and Solaris. The effort led to 1) continued development of LibSBML, (increased support of SBML features and added functionality); and 2) supported SBML use and evolution (direct support for DARPA Bio-SPICE).

15. SUBJECT TERMS
systems biology, simulation, robustness, markup languages, Systems Biology Markup Language (SBML)

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT:	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON (Monitor)
a. REPORT	b. ABSTRACT	c. THIS PAGE			James B. Moncrief
Unclassified	Unclassified	Unclassified	SAR	80	19b. TELEPHONE NUMBER *(Include Area Code)* (937) 255-6548 x3606

Contents

List of Figures

List of Tables

1 Project Goals

The overall long-term objective of our research is to develop mathematical and software infrastructure in support of post-genomics research in systems biology. One near-term objective articulated in this abstract centers on a deeper understanding of the organizational principles of biological networks. A distinguishing theme of this work is its focus on scalable methods of robustness and model (in)validation with data, as opposed to relying purely on simulation. In computability terms, if simulation is viewed as a way to attack the NP hard side of biological problems, our approach attacks the coNP side. Much of the success of reductionist biology has depended on creative individuals who draw biologically meaningful inferences from data and computation using small scale and informal reasoning. This type of inference was critical because the reductionist research program itself offered no systematic tools to deal with complexity, only with the component parts. Far from being dispensed with, this reasoning process and its biological content must be both formalized and made rigorous, systematic, and scalable as well, and ultimately teachable. This requires the development of new mathematics as well as algorithms and software.

A central goal of modeling and simulation is to connect molecular mechanisms to network function to questions of biomedical relevance. Unfortunately, many of the most critical questions involve events which are extremely rare at the individual cell level where the mechanisms act yet catastrophic to the organism. Thus simulation methods that may be adequate for studying generic or typical behavior are entirely inadequate to explore such worst-case scenarios, which with conventional methods are computational intractable. We are extending the best-practice tools and algorithms for robustness analysis that have become standards in engineering to models of biological relevance, which are typically nonlinear, hybrid, uncertain, and stochastic. This includes integrating formal inference methods from the previously fragmented theories in Computer Science with those of Control and Dynamical Systems. This involves deep mathematical challenges that parallel those for technological networks, for which we have made dramatic progress, and on which we are building new tools for systems biology.

1.1 Sum of Squares (SOS) Framework and SOSTOOLS

Biological networks connect devices of enormous complexity and sophistication even at their molecular level into modular components for sensing, signal processing, communication, computation, and actuation. These components are further integrated into vast regulatory networks with layers of feedback. No one doubts the vast range of capabilities that a deep understanding of this biological complexity would enable, but beyond the need for improved experimental technology, and sophisticated bionformatics to manage the data such improvements would continue to yield, there is little consensus as to exactly what further must be done. We claim that the overwhelmingly greatest source of complexity and the least current understanding lies in the signaling, communications, and computation modules, and even more so in the feedback control systems that they comprise. The solution to the challenge of biological complexity may not be to fundamentally change the way in which current molecular biology research is done so much as to augment this research with a new way of thinking about the systems integration issue itself.

Despite its enormous success, the reductionist program provides a poor foundation for many new technical challenges. For example, the ubiquitous connectivity and flexibility of the Internet as observed by the user is taken for granted, as are the wires, chips, and displays that make up the hardware, but it is rare for nonexperts to be aware of the complex layers of protocols and feedback regulation that makes the Internet's flexibility and robustness possible. Until recently, there has been limited theoretical support for the study of the systems-level challenges in either

internetworking or biology, and limited academic research. Nevertheless, for some time there has been a widely shared vision there could be universal features of complex systems that can transcend these reductionist decompositions [22,25,132], and provide a unifying integration. Sharp differences have arisen however with regard to exactly what those features are. We believe there is now a clear, compelling, and coherent path emerging from the striking convergence of the three research themes of biology, technology, and mathematics.

First, biologists have provided a detailed description of the components of biological networks, and many organizational principles of these networks are becoming increasingly apparent. Second, advanced information technologies have enabled engineering systems to approach biology in their complexity. We are developing new theories that elucidate these similarities that are comparable in depth and richness with those available for more traditional subdisciplines. While these share with their traditional counterparts many of the domain-specific assumptions that overcome the intractability of more general formulations, this progress has sharpened the mathematical questions that are relevant to these important application domains. Thus we have the beginnings of the first coherent, complete theoretical foundation of the Internet [64,65,74,75,121,133], and have also been developing new theory and software infrastructure to support systems biology [25, 31, 34, 45, 57, 130]. We are making rigorous and precise the notion that this apparent network-level evolutionary convergence within and between biology and technology is not accidental, but follows necessarily from the universal requirements of efficiency and robustness.

While the full consequences of the claimed convergence emerging from these two areas will take years to be fully resolved, an important message is now clear. The method of decomposing complex systems into vertical layers of varying complexity and scale, wherein each layer is further decomposed horizontally into modules, appears to be not only ubiquitous but necessary. It is neither an accident of evolution nor merely an artificial construct imposed by humans to make biology and technology comprehensible, although that may be a wonderfully serendipitous side-effect. Thus we do not advocate abandoning the reductionist program of decomposing complexity, but in managing the process more consciously and systematically. The disciplinary decompositions that exist may indeed be historical artifices, but the need for such decompositions is not. The key to creating an integrated approach to understanding, exploiting, and mimicking biological complexity is not to replace existing technologies, but to augment them with a more flexible and rigorous technology for decomposition and recomposition.

Finally, the mathematical foundation is being developed for a far more unified theory of complex systems that overcomes the intractability that forced the disciplinary fragmentation in the first place, and this is the most important development for this project. It is in retrospect unsurprising that a genuinely new science of complexity, particularly biological, would require equally new mathematics to answer basic universal questions such as: Is a model consistent with experimental data, which may come from extremely heterogeneous sources? If so, is it robust to additional perturbations that are plausible but untested? Are different models at multiple scales of resolution consistent? What is the most promising experiment to refute or refine a model? These questions are all naturally nonlinear, nonequilibrium, uncertain, hybrid and so on, and their analysis has relied mainly on simulation. Unfortunately, simulation alone is inadequate. One computer simulation produces one example of one time history for one set of parameters and initial conditions. Thus simulations can only ever provide counterexamples to hypotheses about the behavior of a complex system, and can never provide *proofs*. (In technical terms, they can in principle provide satisfactory solutions to questions in NP, but not to questions in coNP.) Simulations can never prove that a given behavior or regularity is necessary and universal; they can at best show that a behavior is generic or typical. What is needed is an effective (and scalable) method for, in essence, systematically proving robustness properties of nonlinear dynamical systems. The possibility of such a thing (especially

without P=NP=coNP in computational complexity theory) is profound and remarkable, and it is the foundation of our approach.

1.1.1 The Organization of Biological Networks

One of our goals is to develop a theory of biological organization that exploits the features of evolution and robustness to constrain the search spaces in our analysis algorithms. Specifically, our computational methods for modeling from data, simulation, and robustness analysis need not solve arbitrary unstructured problems, which are certainly intractable, but only those that are biologically meaningful. Biological systems at every level of organization are highly structured, far from equilibrium, persist there robustly despite fluctuations in their environment and their components, and have evolved to this highly organized state. This places constraints on biological organization that has some parallels in technology but none in the other sciences. Algorithms that exploit this organization can be almost arbitrarily more efficient and reliable than those that do not, but it requires a rigorous theory to connect the robustness and evolvability of biological networks, with algorithms for modeling and system identification, analysis, and simulation. All of our results so far are extremely encouraging, but are merely the beginning of what we believe is possible.

Our main effort on organizational principles is to identify the features of biological networks, as opposed to arbitrary sets of chemical reactions, that make automatic and scalable computational methods feasible, even when the computational complexity classes are worst-case exponential or worse. A smaller effort has been focused on additionally elucidating organizational principles to provide greater understanding of biological complexity. We want to help answer the question "what is all this complexity for?" [59]. This will have a huge impact on computation, but the results can go beyond that. In particular, our program could be viewed as thinking of biological networks as a kind of technological network built on the physical substrate of biochemistry, as opposed to, say, the CMOS VLSI and fiber optics of the Internet. Biological networks integrate controls, communications, and computing in a way that engineers are just beginning to understand in a deeply theoretical way, and we have had great success on the forefront of those efforts. By explicitly connecting the theoretical challenges in advanced technological and biological networks, there is the promise for substantial synergy, and there is strong evidence already that this approach will bring novel insights to both areas [2, 24, 29, 55, 62, 111, 118].

1.1.2 Robust yet fragile systems

An emphasis on scalable and provably correct analysis methods is not just for mathematical completeness, but is driven by a ubiquitous property of complex engineering and natural networked systems: they are *robust yet fragile (RYF)*. Complex networks can provide remarkable robustness despite large perturbations in their environments and component parts, but they can also be extremely fragile to cascading failure events triggered by relatively small perturbations. We experience various illnesses, crashes due to software bugs, viruses, worms, and denial-of-services attacks, power glitches, security screenings, etc, as annoying but rarely catastrophic. Typically, our networks protect us. But cancer and other epidemics, chronic auto-immunity, market crashes, terrorist attacks, large power outages and fires, etc, remind us that our complexity has a price. Indeed, most dollars and lives lost in natural and technological disasters happen in the few largest events, while the typical event is so small as to usually go unreported.

Many current military technical visions convincingly suggests that network complexity can provide robustness and efficiency that ultimately greatly exceeds that of comparable brute force approaches. The ultimate challenge will not be to make this apparent in demonstrations and typical

scenarios, but to avoid the rare but catastrophic failures that seem to inevitably accompany new levels of complexity. Unfortunately, the entire scientific enterprise of experimentation, modeling, and simulation of complex systems has been most successful at studying their typical or generic behaviors. Thus it should be no surprise that the rigorous study of the fragility of complex systems would require new methods.

That the intrinsically "robust yet fragile" (RYF) nature of complex systems [20–22, 29, 100, 132] has the computational counterpart of "dual complexity implies primal fragility" is a key feature of our approach. Practically speaking, this completely changes what is possible computationally. Organisms, ecosystems, and successful advanced technologies are highly constrained in that they are not evolved/designed arbitrarily, but necessarily in ways that are robust to uncertainties in their environment and their component parts. These are extremely severe constraints, not present in other sciences but essential in both biology and engineering. The most obvious feature is that their macroscopic system properties can be both extremely robust to most microscopic details yet hyper-fragile to a few, and this must shape both modeling and analysis, and the experimental process that it interacts with. If most details don't matter, most experiments are relatively uninformative. If a few details are crucial, then this is where both modeling and experiments must focus, but neither a purely top-down nor bottom-up approach can reliably find them.

Thus failure to explicitly exploit the highly structured, organized, and "robust yet fragile" nature of such systems hopelessly dooms any method to be overwhelmed by their sheer complexity. Technically speaking, we can now formulate a wide range of questions for very general dynamical systems under a common Lyapunov-type umbrella, converting them into statements involving semi-algebraic sets, polynomial (nonlinear) equations and inequalities. Proving such statements is still coNP-hard, but real algebraic geometry, semi-definite programming, and duality theory from optimization provide new methods to systematically exhaust coNP by searching for nested families of short proofs using convex relaxations. Not only can we search for short proofs systematically, but a lack of short proofs implies, by a generalization of duality, intrinsic fragilities in the question itself. This feedback from computation to modeling does not imply P=NP=coNP, which is unlikely, but rather that inference problems within coNP lacking short proofs can be traced to specific and meaningful flaws in models or data for which resolution can then be systematically pursued. Note that this is a radical broadening of the numerical analysts notion of ill-conditioning, and involves mathematics from a variety of previously unrelated disciplines. Again, in retrospect, this should not be surprising, but it creates enormous challenges in both education and the review process.

Though this is all very new, these methods have already found substantial applications in networking, biology, physics, dynamical systems, controls, algorithms, and finance [10–13, 28, 34, 41, 78, 81, 89–91, 98, 124–128], and work on connections with communications theory is in progress and discussed in the technical details. A side benefit of a deepening understanding of the fundamental nature of complexity in a general sense is also a new and more rigorous explanations for long-standing problems in physics associated with complex systems [5, 10–13, 21, 28, 28, 66, 97, 117].

1.1.3 Robust and Scalable Validation of Models Against Data

Simulation will always be a workhorse of systems biology, but it can be enhanced substantially if conjectures formulated using simulation can be proved rigorously. The linchpin of our proposed modeling system is the development and implementation of theoretically-sound methods for model validation. Although some existing software tools provide mechanisms for comparing a model's behavior to experimental data (e.g., Gepasi [68, 69]), the methods used to date have been ad-hoc, brute-force approaches that do not scale to larger models. The theoretical framework described later in this document represents an unprecedented opportunity to create a system for model analysis

and validation and iterative experimentation for large-scale, stochastic, nonlinear, nonequilibrium, mixed continuous and discrete models with multiple time and spatial scales. The remarkable quality of the theory is that it can be used to prove conjectures for this difficult class of models such as "this model cannot fit the data, no matter what parameters we use" and "this model is robust no matter how parameters are varied." This is something that previously has not been possible except for much simpler models. Yet this, together with sophisticated robustness analysis methods, is exactly the capability needed in order to allow realistic biological models to be analyzed and related back to the experimental data to help answer the question, "What is the next experiment that would best differentiate between the current alternative hypotheses?"

We are relying on SOSTOOLS as a foundation for the system identification and parameter estimation research, however this reliance is less than it might appear. Our SOS/SDP framework actually recovers as special cases essentially all of the standard methods, so the worst-case scenario is that it merely provide an integrated and unified method to access what might otherwise appear to be quite disparate methods. This is not an aspect of our methods that we emphasize but it is an important element in our optimism about their potential. Perhaps more important is the converse, that we are suspicious of new methods that cannot capture gold-standard methods already in existence. Another important issue is that the SOS/SDP methods are the only candidates for a successor to linear programs in providing all the features of automation, scalability, duality, structure, and fragility to hard problem classes involving stochastic, hybrid, and nonlinear dynamical systems, in addition to reducing to linear programs in the special cases when it applies. These two features, plus the implementation in a MATLAB toolbox, makes SOSTOOLS unique. It also has the benefit that a large, diverse, and sophisticated research community spanning control and dynamical systems, hybrid systems, optimization, and many areas of pure and applied mathematics has recently begun to focus substantial attention in this area.

1.2 SBML

In order to perform computational analysis on models of biological systems, it is first necessary to represent those models in a computable format. This format should be common to all the software tools (editors, simulators, analysis algorithms, and databases) used to perform computational modeling and analysis. The Systems Biology Markup Language (SBML) project is aimed at addressing exactly this need for biochemical network modeling by creating a lingua franca for computational models—a common format for communicating the most essential aspects of models between software tools. Our team developed SBML into a de facto standard format for representing formal qualitative and quantitative models of systems of biochemical reactions. It has become an unparalleled success in this area, supported by over 110 software tools worldwide to date.

SBML is a machine-readable model definition language based upon XML, the eXtensible Markup Language [15,18], which in turn is a simple and portable text-based substrate that has been gaining widespread acceptance in computational biology and bioinformatics. Software tools read and write models in SBML by using either a general-purpose XML parsing library (two popular ones are Xerces and Expat), or a specialized library that provides a higher-level interface to working with SBML. Software developers who start with general-purpose XML libraries find themselves having to write constructs that abstract above the level of raw XML and provide more specialized (i.e., SBML-specific) functionality anyway; thus, for most developers, the second approach of starting with a specialized SBML library is much more efficient and desirable. To this end, our team has developed *libSBML*, an embedded library for working with SBML (*libSBML*).

The *libSBML* library provides an interface for working with SBML in a number of programming languages: C, C++, Java, Perl, MATLAB and Python. It is free, open-source, and portable

to Linux, Windows, MacOS and Solaris. It also provides many important capabilities, such as consistency-checking of SBML models, translation of mathematical expressions, and others. The *libSBML* library is one of the most popular programming tools for working with SBML, and has proven to be so robust that The MathWorks (makers of MATLAB) use it as part of their commercial SimBiology package for MATLAB.

This project's goals were in two areas: (A) continued development on *libSBML*, both to complete its support of SBML and to add important additional functionality; and (B) support of SBML use and evolution in the context of specific BioSPICE project needs.

1.2.1 Motivations for Developing SBML

Until recently, the majority of models were implemented in custom programs and published as statements of the underlying mathematics (e.g., sets of equations). However, to be useful as formal embodiments of our understanding of biological systems [16], computational models must be put into a consistent form that can be communicated more directly between the software tools used to work with them. This format must help overcome a number of problems facing a systems biologist:

- Users often need to work with complementary resources from multiple software tools in the course of a project because different tools have different strengths and capabilities. For example, one tool may have a good model editing interface, another tool may provide novel facilities for analyzing system properties, yet another may implement an advanced simulation capability but lack a good graphical interface, etc. If the tools do not share a common model storage format, users are forced to re-encode their models in each tool separately, a time-consuming and error-prone practice.

- Models published in peer-reviewed journals are sometimes accompanied by instructions for obtaining the definitions in electronic form. However, because each author may use a different software environment (and associated model representation language), such definitions are often not straightforward to examine, test and reuse. Researchers who wish to use a published model typically must transcribe it manually into a format compatible with their particular software.

- When simulation software packages are no longer supported, models developed in those systems can become stranded and unusable. This has already happened on a number of occasions, with a resulting loss of usable models to the research community. Continued innovation and development of new tools will only aggravate this problem unless the issue of standard formats is addressed.

- Reuse of existing models requires that those models can be clearly identified, easily retrieved, and related to their published descriptions in the scientific literature. Moreover, because of the increasing size and complexity of models continually being developed, the model structure should be documented to allow for efficient handling and sound modification.

We developed the Systems Biology Markup language (SBML) in an effort to address these problems. SBML is a format for representing computational models in a way that can be used by different software systems to communicate and exchange those models [38, 50, 51]. By supporting SBML as an input and output format, different software tools can all operate on an identical representation of a model, removing opportunities for errors in translation and assuring a common starting point for analyses and simulations.

1.2.2 The Form and Features of SBML

SBML can encode models consisting of biochemical entities (species) linked by reactions to form networks. An important principle is that models are decomposed into explicitly-labeled constituent elements, the set of which resembles a verbose rendition of chemical reaction equations. The representation deliberately does not cast the model directly into a set of differential equations or other specific interpretation of the model. This explicit, modeling-framework-agnostic decomposition makes it easier for a software tool to interpret the model and translate the SBML form into whatever internal form the tool actually uses. The formalisms in SBML allows a wide range of biological phenomena to be modeled, including metabolism, cell signaling, gene regulation, and more. Significant flexibility and power comes from the ability to define arbitrary formulae for the rates of change of variables as well as the ability to express other constraints mathematically.

Balancing simplicity and feature-richness has been challenging because of the variety of software tools and techniques being explored in the field, and because it is difficult to predict how all language design choices will ultimately impact software and users. SBML development has therefore proceeded in a staged approach where each SBML *Level* is an attempt to achieve a consistent language that supports the needs of a significant number of software tools at a certain level of complexity. *Versions* within levels refine and adjust language features. This staged development provides software authors with stable standards so the community can gain experience with the language definitions before new features are introduced. The evolution towards feature stability within a given Level is an iterative process requiring much discussion with the user community. Eventually, the community decides that it is time to stop refining a given SBML Level, take a leap in complexity and move to developing the next higher level. Two levels have been finalized so far: Level 1 and Level 2 [37, 51]. The former is simpler (but less powerful) than Level 2. The separate levels are intended to coexist; SBML Level 2 does not render Level 1 obsolete. Software tools that do not need or cannot support higher levels can go on using lower levels; tools that can read higher levels are assured of also being able to interpret models defined in the lower levels. The *libSBML* open-source software infrastructure we have been developing allows developers to support both Levels in their software almost transparently.

SBML Level 2 Version 2 is the latest definition of SBML; it is an incremental evolution of the language resulting from the practical experiences of users and developers working with Level 1 and Level 2 Version 1. A model definition in Level 2 Version 2 consists of lists of one or more of the following components: *species type*, a type of entity that can be involved in reactions (e.g., molecules, ions, proteins); *compartment type*, a type of location where reacting entities such as chemical substances may be located; *compartment*, a container for well-stirred substances where reactions take place; *species*, a pool of a chemical substance located in a specific compartment (a species represents the concentration or amount of a substance and not a single molecule); *reaction*, a statement describing some transformation, transport or binding process that can change one or more species (each reaction is characterized by the stoichiometry of its products and reactants and optionally by a rate equation); *parameter*, a quantity that has a symbolic name; *unit definition*, a name for a unit used in the expression of quantities in a model; *initial assignment*, a mathematical expression used to determine the initial value of a variable; *constraint*, a mathematical expression that defines a constraint on the values of model variables throughout simulation time; *rule*, a mathematical expression that is added to the model equations constructed from the set of reactions (rules can be used to set parameter values, establish constraints between quantities, etc.); *function*: a named mathematical function that can be used in place of repeated expressions in rate equations and other formulas; and *event*: a set of mathematical formulae evaluated at a specified moment in the time evolution of the system. Additional features in SBML Level 2 include support for

including metadata using RDF annotations [61], and support for delay functions.

1.2.3 Relationships to Other Representation Languages

Many XML-based formats exist for representing data and models in different domains of biology, for example MAGE-ML [110] and PSI-MI [49], and others. However, we know of only two formats that are suitable for representing compartmental reaction network models with sufficient mathematical depth that the descriptions can be used as direct input to simulation software. The two are SBML and CellML [47, 48].

CellML is based on composing systems of equations by linking together the variables in those equations; it also has features for declaring biochemical reactions explicitly, as well as encapsulating arbitrary components into modules. Its focus is a component-based architecture to facilitate reuse of models and parts of models, and the mathematical description of models. By contrast, SBML provides constructs that are more similar to the internal data objects used in many contemporary software packages specialized for biochemical networks. These differences aside, the SBML and CellML efforts share much in common and represent somewhat different approaches to solving similar problems. They were initially developed independently, but the developers of both languages continue to exchange ideas and seek ways of making the languages more interoperable. SBML Level 2 borrows ideas from CellML.

A recent related effort is BioPAX. The BioPAX (*Biological Pathway Exchange*; [9,114]) project is an effort to create a data exchange format for biological pathway data. Much like SBML, it aims to be a lingua franca for pathway databases, an area currently lacking a standard format despite the existence of well over 100 Internet-accessible databases storing pathway data. BioPAX deliberately borrows ideas from SBML development [4], such as the concept of stratifying development into levels, and is in many ways complementary to SBML.

1.2.4 LibSBML: Software Infrastructure for SBML

LibSBML is an application programming interface (API) library for reading, writing and manipulating files and data streams containing SBML content. Developers can embed the library in their applications, saving themselves the work of implementing their own parsing, manipulation and validation software. At the API level, the library provides the same interface to data structures independently of whether the model originated in SBML Level 1 or 2. The library currently also offers the ability to translate SBML Level 1 models to SBML Level 2.

LibSBML is written in ISO standard C and C++ and is highly portable. It is currently supported on the Linux, Solaris, MacOS X, and Microsoft Windows operating systems. The library provides language bindings for C, C++, Java, Python, Perl, MATLAB and Common Lisp, with support for other languages planned for the future. We distribute the package in both source-code form and as precompiled dynamic libraries for the Microsoft Windows, Linux and Apple MacOS X operating systems; they are available under terms of the LGPL [39] from the *sbml* project on SourceForge.net [109], the world's largest open-source software repository and project hosting service. LibSBML is at release version 2.3.2 as of June 2005.

1.2.5 Advantages of a Dedicated Library for SBML

Why not simply use a generic XML parsing library? After all, SBML is usually expressed in XML, and there exist plenty of XML parsers, so why not simply tell people to use one of them, rather than develop a specialized library? The answer is: while it is true that developers *can* use general-

purpose XML libraries, there are many reasons why using a system such as libSBML is a vastly better choice.

One of the features of libSBML is its facilities for manipulating mathematical formulas supporting differences in representation between SBML Level 1 and SBML Level 2. As discussed in more detail below, libSBML provides an API that allows working with formulas in both text-string and MathML [3] form, and to interconvert mathematical expressions between these forms. The utility of this facility extends beyond converting between SBML Level 1 and 2. Many software packages provide users with the ability to express formulas for such things as reaction rate expressions, and these packages' interfaces often let users type in the formulas directly as text strings. LibSBML saves application programmers the work of developing formula manipulation and translation functionality. It makes it possible to translate those formula strings directly into Abstract Syntax Trees (ASTs), manipulate them using AST operations, and write them out in the MathML format of SBML Level 2.

As discussed below, another feature of libSBML is the validation it performs on SBML inputs at the time of parsing files and data streams. This helps verify the correctness of models in a way that goes beyond simple syntactic validation. Still another invaluable feature of libSBML is the domain-specific operations it provides beyond simple SBML-specific accessor facilities. Examples of such operations include obtaining a count of the number of boundary condition species, determining the modifier species of a reaction (assuming the reaction provides kinetics), and constructing the stoichiometric matrix for all reactions in a model.

Finally, libSBML is solidly written and tested. The entire library has been written by seasoned, professional, software engineers using the test-driven approach [6]. The libSBML source code currently has 760 unit tests and over 3400 individual assertions. It represents a robust and well-tested system that others can build upon.

2 Approach/Problems to be Addressed

The following provide more details about the areas of work that were the goals for this funding.

2.1 Sum of Squares (SOS) Framework and SOSTOOLS

Consider a given system of polynomial equations and inequalities, for instance:

$$\begin{aligned} f_1(x_1, x_2) &:= \quad x_1^2 + x_2^2 - 1 = 0, \\ g_1(x_1, x_2) &:= \quad 3\,x_2 - x_1^3 - 2 \geq 0, \\ g_2(x_1, x_2) &:= \quad x_1 - 8\,x_2^3 \geq 0. \end{aligned} \tag{1}$$

How can one find real solutions (x_1, x_2)? How to *prove* that they do not exist? And if the solution set is nonempty, how to optimize a polynomial function over this set?

Until a few years ago, the default answer to these and similar questions would have been that the possible nonconvexity of the feasible set and/or objective function precludes any kind of analytic global results. Even today, the methods of choice for most practitioners would probably employ mostly local techniques (Newton's and its variations), possibly complemented by a systematic search using deterministic or stochastic exploration of the solution space, interval analysis or branch and bound. However, very recently there have been renewed hopes for the efficient solution of specific instances of this kind of problems. The main reason is the appearance of methods that combine in a very interesting fashion ideas from *real algebraic geometry* and *convex optimization* [60, 71, 78]. As

we will see, these methods are based on the intimate links between sum of squares decompositions for multivariate polynomials and semidefinite programming (SDP).

We will outline the essential elements of this new research approach as introduced in [78, 79]. The centerpieces will be the following two facts about multivariate polynomials and systems of polynomials inequalities:

1. Sum of squares decompositions can be computed using semidefinite programming.

2. The search for infeasibility certificates is a convex problem. For bounded degree, it is an SDP.

We will define the basic ideas needed to make the assertions above precise, and explain the relationship with earlier techniques. For this, we will introduce sum of squares polynomials and the notion of *sum of squares programs*. We then explain how to use them to provide infeasibility certificates for systems of polynomial inequalities, finally putting it all together via the surprising connections with optimization.

2.1.1 Sums of Squares and SOS Programs

Our notation is mostly standard. The *monomial* x^α associated to the n-tuple $\alpha = (\alpha_1, \ldots, \alpha_n)$ has the form $x_1^{\alpha_1} \ldots x_n^{\alpha_n}$, where $\alpha_i \in \mathbb{N}_0$. The *degree* of a monomial x^α is the nonnegative integer $\sum_{i=1}^n \alpha_i$. A *polynomial* is a finite linear combination of monomials $\sum_{\alpha \in S} c_\alpha x^\alpha$, where the coefficients c_α are real. If all the monomials have the same degree d, we will call the polynomial *homogeneous* of degree d. We denote the ring of multivariate polynomials with real coefficients in the indeterminates $\{x_1, \ldots, x_n\}$ as $\mathcal{R}[x]$.

A multivariate polynomial is a *sum of squares* (SOS) if it can be written as a sum of squares of other polynomials, i.e.,

$$p(x) = \sum_i q_i^2(x), \quad q_i(x) \in \mathcal{R}[x].$$

If $p(x)$ is SOS then clearly $p(x) \geq 0$ for all x. In general, SOS decompositions are not unique. For example, the polynomial $p(x_1, x_2) = x_1^2 - x_1 x_2^2 + x_2^4 + 1$ is SOS. Among infinite others, it has the decompositions: $p(x_1, x_2) = \frac{3}{4}(x_1 - x_2^2)^2 + \frac{1}{4}(x_1 + x_2^2)^2 + 1 = \frac{1}{9}(3 - x_2^2)^2 + \frac{2}{3}x_2^2 + \frac{1}{288}(9x_1 - 16x_2^2)^2 + \frac{23}{32}x_1^2$. The sum of squares condition is a quite natural sufficient test for polynomial nonnegativity. Its rich mathematical structure has been analyzed in detail in the past, notably by Reznick and his coauthors [23, 101], but until very recently the computational implications have not been fully explored. In the last few years there have been some very interesting new developments surrounding sums of squares, where several independent approaches have produced a wide array of results linking foundational questions in algebra with computational possibilities arising from convex optimization. Most of them employ semidefinite programming (SDP) as the essential computational tool. For completeness, we present in the next paragraph a brief summary of SDP.

Semidefinite programming SDP is a broad generalization of linear programming (LP), to the case of symmetric matrices. Denoting by $\mathcal{S}^n$ the space of $n \times n$ symmetric matrices, the standard SDP primal-dual formulation is:

$$\begin{array}{lll} \min_X \; C \bullet X & \text{s.t.} & \left\{ \begin{array}{l} A_i \bullet X = b_i, \;\; i = 1, \ldots, m \\ X \succeq 0 \end{array} \right. \\ \max_y \; b^T y, & \text{s.t.} & \sum_{i=1}^m A_i y_i \preceq C \end{array} \tag{2}$$

where $A_i, C, X \in \mathcal{S}^n$ and $b, y \in \mathcal{R}^m$. The matrix inequalities are to be interpreted in the partial order induced by the positive semidefinite cone, i.e., $X \succeq Y$ means that $X - Y$ is a positive semidefinite matrix. Since its appearance almost a decade ago (related ideas, such as eigenvalue optimization, have been around for decades) there has been a true "revolution" in computational methods, supported by an astonishing variety of applications. By now there are several excellent introductions to SDP; among them we mention the well-known work of Vandenberghe and Boyd [120] as a wonderful survey of the basic theory and initial applications, and the handbook [122] for a comprehensive treatment of the many aspects of the subject.

From SDP to SOS The main object of interest in semidefinite programming is quadratic forms, that are positive semidefinite. When attempting to generalize this construction to homogeneous polynomials of higher degree, an unsurmountable difficulty that appears is the fact that deciding nonnegativity for quartic or higher degree forms is an NP-hard problem. Therefore, a computational tractable replacement for this is the following: even degree polynomials, that are sums of squares.

Sum of squares programs can then be defined as optimization problems over affine families of polynomials, subject to SOS contraints. Like SDPs, there are several possible equivalent descriptions. We choose below a free variables formulation, to highlight the analogy with the standard SDP dual form discussed above. A sum of squares program has the form

$$\begin{array}{rl} \max_y & b_1 y_1 + \cdots + b_m y_m \\ \text{s.t.} & P_i(x,y) \text{ are SOS}, \quad i = 1, \ldots, p \end{array}$$

where $P_i(x,y) := C_i(x) + A_{i1}(x)y_1 + \cdots + A_{im}(x)y_m$, and the C_i, A_{ij} are given polynomials in the variables x_i.

SOS programs are very useful, since they directly operate with polynomials as their basic objects, thus providing a quite natural modelling formulation for many problems. Among others, examples for this are the search for Lyapunov functions for nonlinear systems [76, 78], probability inequalities [7], as well as the relaxations in [60, 78] discussed below.

Interestingly enough, despite their apparently greater generality, sum of squares programs are in fact *equivalent* to SDPs. On the one hand, by choosing the polynomials $C_i(x), A_{ij}(x)$ to be quadratic forms, we recover standard SDP. On the other hand, as we will see in the next section, it is possible to exactly embed every SOS program into a larger SDP. Nevertheless, the rich algebraic structure of SOS programs will allow us a much deeper understanding of their special properties, as well as enable customized, more efficient algorithms for their solution [63]. Furthermore, as illustrated in later sections, there are numerous questions related to some foundational issues in nonconvex optimization that have simple and natural formulations as SOS programs.

SOS programs as SDPs Sum of squares programs can be written as SDPs. The reason is the following theorem: A polynomial $p(x)$ is SOS if and only if $p(x) = z^T Q z$, where z is a vector of monomials in the x_i variables, $Q \in \mathcal{S}^N$ and $Q \succeq 0$. In other words, every SOS polynomial can be written as a quadratic form in a set of monomials of cardinality N, with the corresponding matrix being positive semidefinite. The vector of monomials z (and therefore N) in general depends on the degree and sparsity pattern of $p(x)$. If $p(x)$ has n variables and total degree $2d$, then z can always be chosen as a subset of the set of monomials of degree less than or equal to d, of cardinality

$N = \binom{n+d}{d}$. Consider again the polynomial $p(x_1, x_2) = x_1^2 - x_1 x_2^2 + x_2^4 + 1$. It has the representation

$$p(x_1, x_2) = \frac{1}{6} \begin{bmatrix} 1 \\ x_2 \\ x_2^2 \\ x_1 \end{bmatrix}^T \begin{bmatrix} 6 & 0 & -2 & 0 \\ 0 & 4 & 0 & 0 \\ -2 & 0 & 6 & -3 \\ 0 & 0 & -3 & 6 \end{bmatrix} \begin{bmatrix} 1 \\ x_2 \\ x_2^2 \\ x_1 \end{bmatrix},$$

and the matrix in the expression above is positive semidefinite.

In the representation $f(x) = z^T Q z$, for the right- and left-hand sides to be identical, all the coefficients of the corresponding polynomials should be equal. Since Q is simultaneously constrained by linear equations and a positive semidefiniteness condition, the problem can be easily seen to be directly equivalent to an SDP feasibility problem in the standard primal form (2). Given a SOS program, we can use the theorem above to construct an equivalent SDP. The conversion step is fully algorithmic, and has been implemented, for instance, in the SOSTOOLS software package, described in the next section. Therefore, we can in principle directly apply all the available numerical methods for SDP to solve SOS programs.

2.1.2 Algebra and Optimization

A central theme throughout convex optimization is the idea of *infeasibility certificates* (for instance, in LP via Farkas' lemma), or equivalently, *theorems of the alternative*. As we will see, the key link relating algebra and optimization in this approach is the fact that infeasibility can *always* be certified by a particular algebraic identity, whose solution is found via convex optimization.

Ideals and cones For later reference, we define here two important algebraic objects: the *ideal* and the *cone* associated with a set of polynomials: Given a set of multivariate polynomials $\{f_1, \ldots, f_m\}$, let

$$\textbf{ideal}(f_1, \ldots, f_m) := \{f \mid f = \sum_{i=1}^{m} t_i f_i, \quad t_i \in \mathcal{R}[x]\}.$$

Also, given a set of multivariate polynomials $\{g_1, \ldots, g_m\}$, let

$$\textbf{cone}(g_1, \ldots, g_m) := \{g \mid g = s_0 + \sum_{\{i\}} s_i g_i + \sum_{\{i,j\}} s_{ij} g_i g_j + \sum_{\{i,j,k\}} s_{ijk} g_i g_j g_k + \cdots\},$$

where each term in the sum is a squarefree product of the polynomials g_i, with a coefficient $s_\alpha \in \mathcal{R}[x]$ that is a sums of squares. The sum is finite, with a total of $2^m - 1$ terms, corresponding to the nonempty subsets of $\{g_1, \ldots, g_m\}$.

These algebraic objects will be used for deriving *valid inequalities*, which are logical consequences of the given constraints. Notice that by construction, every polynomial in $\textbf{ideal}(f_i)$ vanishes in the solution set of $f_i(x) = 0$. Similarly, every element of $\textbf{cone}(g_i)$ is clearly nonnegative on the feasible set of $g_i(x) \geq 0$. The notions of *ideal* and *cone* as used above are standard in real algebraic geometry; see for instance [14]. In particular, the cones are also referred to as a *preorders*. Notice that as geometric objects, ideals are affine sets, and cones are closed under convex combinations and nonnegative scalings (i.e., they are actually cones in the convex geometry sense). These convexity properties, coupled with the relationships between SDP and SOS, will be key for our developments in the next section.

Infeasibility certificates If a system of equations does not have solutions, how do we *prove* this fact? A very useful concept is that of *certificates*, which are formal algebraic identities that provide irrefutable evidence of the nonexistence of solutions.

We briefly illustrate some well-known examples below. The first two deal with linear systems and polynomial equations over the complex numbers, respectively.

- Range/kernel: $Ax = b$ is infeasible $\Leftrightarrow \exists\, \mu$ s.t. $A^T\mu = 0,\ b^T\mu = -1$.
- Hilbert's Nullstellensatz: Let $f_i(z), \ldots, f_m(z)$ be polynomials in complex variables $z_1, \ldots, z_n$. Then,

$$f_i(z) = 0 \quad (i = 1, \ldots, m) \quad \text{is infeasible in } \mathbb{C}^n \Leftrightarrow -1 \in \mathbf{ideal}(f_1, \ldots, f_m).$$

Each of these theorems has an "easy" direction. For instance, for the first case, given the multipliers μ the infeasibility is obvious, since

$$Ax = b \quad \Rightarrow \quad \mu^T Ax = \mu^T b \quad \Rightarrow \quad 0 = -1,$$

which is clearly a contradiction. The two theorems above deal only with the case of *equations*. The inclusion of inequalities in the problem formulation poses additional algebraic challenges, because we need to work on an *ordered* field. In other words, we need to take into account special properties of the *reals*, and not just the complex numbers.

For the case of *linear* inequalities, LP duality provides the following characterization (Farkas lemma):

$$\left\{ \begin{array}{rcl} Ax + b & = & 0 \\ Cx + d & \geq & 0 \end{array} \right. \text{ is infeasible } \Leftrightarrow \exists\, \lambda \geq 0,\, \mu \text{ s.t. } \left\{ \begin{array}{rcl} A^T\mu + C^T\lambda & = & 0 \\ b^T\mu + d^T\lambda & = & -1. \end{array} \right.$$

Although not widely known in the optimization community until recently, it turns out that similar certificates do exist for *arbitrary* systems of polynomial equations and inequalities over the reals. The result essentially appears in this form in [14], and is due to Stengle [112], and is called Positivstellensatz.

$$\left\{ \begin{array}{rcll} f_i(x) & = & 0, & (i = 1, \ldots, m) \\ g_i(x) & \geq & 0, & (i = 1, \ldots, p) \end{array} \right. \text{ is infeasible in } \mathcal{R}^n$$

$$\Updownarrow$$

$$\exists\, F(x), G(x) \in \mathcal{R}[x] \text{ s.t. } \left\{ \begin{array}{l} F(x) + G(x) = -1 \\ F(x) \in \mathbf{ideal}(f_1, \ldots, f_m) \\ G(x) \in \mathbf{cone}(g_1, \ldots, g_p). \end{array} \right.$$

The theorem states that for every infeasible system of polynomial equations and inequalities, there exists a simple algebraic identity that directly certifies the nonexistence of real solutions. By construction, the evaluation of the polynomial $F(x) + G(x)$ at any feasible point should produce a nonnegative number. However, since this expression is identically equal to the polynomial -1, we arrive at a contradiction. Remarkably, the Positivstellensatz holds under *no assumptions* whatsoever on the polynomials.

In the worst case, the degree of the infeasibility certificates $F(x), G(x)$ could be high (of course, this is to be expected, due to the NP-hardness of the original question). In fact, there are a few explicit counterexamples where large degree refutations are necessary [44]. Nevertheless, for many problems of practical interest, it is often the case that it is possible to prove infeasibility using

Degree \ Field	Complex	Real
Linear	*Range/Kernel* Linear Algebra	*Farkas Lemma* Linear Programming
Polynomial	*Nullstellensatz* Bounded degree: Linear Algebra Groebner bases	*Positivstellensatz* Bounded degree: SDP

Table 1: Infeasibility certificates and associated computational techniques.

relatively low-degree certificates. There is significant numerical evidence that this is the case, as indicated by the large number of practical applications where SDP relaxations based on these techniques have provided solutions of very high quality.

Of course, we are concerned with the effective computation of these certificates. For the Positivstellensatz, notice that the cones and ideals as defined above are always *convex sets* in the space of polynomials. A key consequence is that the conditions in Posivstellensatz for a certificate to exist are therefore *convex*, regardless of any convexity property of the original problem. Even more, the same property holds if we consider only bounded-degree sections, i.e., the intersection with the set of polynomials of degree less than or equal to a given number D. In this case, the conditions in the P-satz have *exactly* the form of a SOS program! Of course, as discussed earlier, this implies that we can find bounded-degree certificates, by solving semidefinite programs. In Table 1 we present a summary of the infeasibility certificates discussed, and the associated computational techniques.

As outlined in the preceding paragraphs, there is a direct connection going from general polynomial optimization problems to SDP, via P-satz infeasibility certificates. Pictorially, we have the following:

$$\text{Polynomial systems} \quad \Rightarrow \quad \text{P-satz certificates} \quad \Rightarrow \quad \text{SOS programs} \quad \Rightarrow \quad \text{SDP}$$

Even though we have discussed only feasibility problems, there are obvious straightforward connections with optimization. By considering the emptiness of the sublevel sets of the objective function, sequences of converging bounds indexed by certificate degree can be directly constructed.

2.1.3 SOSTOOLS

SOSTOOLS [90, 91] is a free, third-party MATLAB toolbox for solving sum of squares programs. The functions implemented in SOSTOOLS are based on the sum of squares decomposition of multivariate polynomials [23], which can be efficiently computed using semidefinite programming [120]. SOSTOOLS was the result of the recent interest in sum of squares polynomials [23, 60, 71, 78, 79, 102, 107], partly due to the fact that these techniques provide convex relaxations for many computationally hard problems such as global, constrained, and boolean optimization [53, 60, 71, 103, 107, 116].

In addition to the optimization problems mentioned above, sum of squares polynomials (and hence SOSTOOLS) find applications in several systems analysis and control theory problems, such as nonlinear stability analysis [76, 78, 89, 104], robustness analysis [34, 76, 78, 89], nonlinear synthesis [52, 93], and model validation [34, 83]. Some other areas in which SOSTOOLS is applicable are geometric theorem proving [80] and quantum physics [28].

Currently, sum of squares programs are handled by reformulating them as semidefinite programs (SDPs), which in turn are solved efficiently, e.g. using interior point methods. Several commercial and non-commercial software packages are available for solving SDPs. While the conversion from SOS programs to SDPs can be performed manually for small size instances or tailored for specific problem classes, such a conversion can be quite cumbersome to perform in general. It is therefore

desirable to have a tool that automatically performs this conversion for general SOS programs. This is exactly where SOSTOOLS comes to play. It automates the conversion from SOS program to SDP, calls the SDP solver, and converts the SDP solution back to the solution of the original SOS program. At present, it uses another free MATLAB add-on called SeDuMi [115] as the SDP solver.

All polynomials in SOSTOOLS are implemented as symbolic objects, making full use of the MATLAB Symbolic Math Toolbox's capabilities. This gives to the user the benefit of being able to do all polynomial manipulations using the usual arithmetic operators: `+`, `-`, `*`, `/`, `^`; as well as differentiation, integration, point evaluation, etc. In addition, this provides the possibility of interfacing with the Maple symbolic engine and library, which is advantageous.

The user interface has been designed to be as simple, easy to use, and transparent as possible. A user creates an SOS program by declaring SOS program variables, adding SOS program constraints, setting the objective function, and so on. After the program is created, the user calls one function to run the solver. Finally, the user retrieves solutions to the SOS program using another function.

SOSTOOLS is available for free under the GNU General Public License. The software and its user's manual can be downloaded from the SOSTOOLS website [91]. It requires MATLAB version 6.0 or later, SeDuMi version 1.05, and the Symbolic Math Toolbox version 2.1.2. SOSTOOLS can be easily run on a UNIX workstation or on a Windows PC. It utilizes the MATLAB sparse matrix representation for good performance and to reduce the amount of memory needed. To give an illustrative figure of the computational load, all the demo files that are distributed along with SOSTOOLS can be solved in less than 8 seconds by SOSTOOLS running on a PC laptop with a 933 Mhz Intel Pentium-III processor and 384 MBytes of RAM.

2.2 SBML

2.2.1 Continued Development of *libSBML*

Expand the Model Consistency Checking Rules and Facilities The SBML specification is defined by an XML Schema that describes the basic syntax and structure of SBML. Unfortunately, the types of data criteria that may be expressed with XML Schemas are somewhat limited and the SBML community has quickly outgrown them. We believe the most elegant, expressive, and scalable form for expressing and validating sophisticated relationships between SBML objects is a programming language that can inspect the *libSBML* object model. As a proof-of-concept, our team has implemented over 30 model consistency checks in *libSBML*. These checks fall into several broad categories, including referential integrity (e.g., is reaction's product defined?), units (e.g., are a compartment's spatial dimensions consistent with its declared units?), and mathematics (e.g., are all symbols in an equation accounted for as either species concentrations or parameters?). We have identified approximately 50 more consistency checks that are important to implement in *libSBML*; a list is available on the SBML project web site's wiki area (`http://sbml.org/wiki`). In addition to implementing the remaining 50 consistency checks, we also planned to add the capability for programmers to extend *libSBML*'s built-in rules with new rules of their own.

Integrate a Units Checking and Translation System An important but undervalued feature of SBML is its support for specifying units on quantities. From the user's standpoint, supplying units allows the user to use those units that are most natural to them and their task; from the software's standpoint, units provide the opportunity to perform an important level of consistency checking on a model. The *libSBML* library currently performs some basic unit checks on a model (e.g., to verify that units used in a model have corresponding definitions), but another aim of this

project was to provide a more powerful facility that can (1) interpret the units used in a model and verify that they are semantically correct and consistent, and (2) support translations of quantities with different units. The first goal is needed to ensure that, for example, complicated rate laws such as

$$\frac{\mathrm{V}_{m1} \cdot \mathrm{RNAP} \cdot \mathrm{RNA}_{nuc}}{\mathrm{K}_{m1} + \mathrm{RNA}_{nuc}} \tag{3}$$

end up with proper units after all parameters and mathematical equations are substituted and expanded. The second goal concerns developing an API that allows calling programs to supply a quantity using one set of units and request that *libSBML* translates it into a quantity having another set of units.

Provide an API for SBML Annotations *Annotations* are a regimented means for allowing software tools to insert additional information into an SBML model. They are defined to be in XML format, and allowed to appear on any model element. Software tools are expected to preserve, as much as possible, annotations included in a model by other tools. Currently, annotations are treated as text strings by *libSBML*. Software developers have requested API functionality allowing the manipulation of annotations at a finer-grained level and more structured fashion. This is currently not provided by *libSBML* because we believe the most workable approach is to provide DOM-like XML handling support for annotations, and not all XML parsers provide a DOM interface. (The Expat parser library does not, and it happens to be one of the most popular XML parsers.) Thus, a goal was to implement a limited set of DOM-like interfaces directly in *libSBML* to support structured manipulation of annotations in SBML.

Provide Additional "Convenience" APIs The *libSBML* library provides a variety of functions that go beyond simply accessing and setting values of model components. For example, there are functions for translating mathematics from text strings into MathML, for getting a list of all chemical substances that have been declared as being boundary conditions on a model, and more. As developers continue using *libSBML*, we receive requests for more functionality. One such request has been for functionality for searching models to find elements having chosen criteria, for example to find all reactions in a model which involve a specific chemical species.

Implement Support for RDF and Metadata in SBML RDF (Resource Description Format) is a standardized means of representing information about resources on the Internet, and is specifically suited to representing metadata in XML data streams. In SBML, RDF is the mechanism for representing metadata (such as references to GO ontology terms). One of the goals of this project was to implement an API in *libSBML* for working with RDF-based metadata in SBML models.

Support "Within-SBML" Translations Some model constructs can be expressed in more than one way in SBML. For example, a model can contain mathematical statements assigning values to named parameters, or the model can be written such that the numerical values themselves appear everywhere in the model where they are used. Some software tools cannot handle models containing assignment statements or other kinds of *Rules*, but *can* handle models that contain the values or other kinds of constructs substituted in. For these tools, the second style of model representation is preferable; the first style of model is inaccessible to them on the face of it, yet could in principle also be accessible to the tool if it had a means for interpreting the model and translating it into a more appropriate form. Therefore, another of our goals was to implement this kind of facility in

libSBML. It was meant to provide mechanisms for doing such things as substituting assignments into a model, "expanding" function definitions and substituting them where they are used in a model (analogous to macro substitution in programming languages), and others.

Support Translating SBML Level 2 down to SBML Level 1 The different SBML levels are intended to coexist, so that software tools with different levels of sophistication can nevertheless exchange some types of models. Level 2 subsumes all the functionality of SBML Level 1, and adds more, so that in principle, a software application supporting SBML Level 2 can also support Level 1. But, not surprisingly, the converse is not always true; a Level 2 model cannot always be expressed in Level 1. It *is* possible to do this for some subset of Level 2 models, because not all models make use of all the advanced features available in Level 2. This means that a significant subset of Level 2 models could be made available to tools which can only read Level 1. The best substrate for implementing a translator is *libSBML*, since it is the most advanced software tool for reading, writing and manipulating SBML.

Support SBML Level 2 Version 2, Expected to Be Introduced in 2005 The SBML community is currently evaluating features to be introduced in a Version 2 specification of SBML Level 2 (which is currently at Version 1). This new version of SBML would correct some misfeatures and add support for certain kinds of constructs that the community has decided are much-needed additions. Our team made a preliminary proposal at the SBML Forum meeting in Heidelberg in October 2004; this proposal was met positively and we expect that a final definition of Version 2 will be made in 2005. It will be essential that *libSBML* supports this new version, and a goal of this project was to make the necessary modifications to *libSBML*.

2.2.2 Support of SBML Use and Evolution

Coordination of Annotations Annotations in SBML are a way for software developers to add information to models in cases where the SBML specification does not provide a structured format for that information. This is being used by many groups to experiment with new additions to SBML. The idea of annotations has been a success, and has lead to requests for a way to organize annotations more formally. As part of this work, we aimed to implement a clearinghouse for SBML annotations and help organize different groups' proposed annotations. The clearinghouse was to be online on the SBML website (`http://sbml.org`) and include: (a) description of a proposed set of annotations; (b) information about the authors of the annotations; (c) the definition of the annotations; (d) use-case examples of how the annotations are meant to be employed; (e) information about software tools using the annotations.

BioPAX and SBML BioPAX is an open file format being proposed for the exchange of biological pathway data. SBML and BioPAX in many ways complement each other: SBML provides more information about the reactions and mathematics behind them, whereas BioPAX provides more information about the entities participating in the reactions. It is natural to seek to link the two formats together. As part of this project, we proposed to work on developing a proposal for linking the SBML and BioPAX representations in a way that would allow software tools to exchange single models containing both SBML and BioPAX content.

3 Accomplishments of the Project

In this section, we describe the results of this project for each of the goals listed in the previous section.

3.1 Towards the Multiscale Simulation of Biochemical Networks

In this section, based on [33], we report on new contributions to the development of a scalable scientific theory and software infrastructure for complex biological networks. We specifically provide a promising approach to model validation, robustness and stochastic reachability analysis, all in the context of two biologically motivated and functionally important systems: the heat shock response in *E. coli* and the lysis/lysogeny decision system in the bacteriophage λ. Both are among the most familiar and widely studied networks in biology. More specifically, using the heat shock response as a case study, we study the important problems of model validation/invalidation and robustness analysis. The analysis of the heat shock system is carried for a deterministic mathematical model that assumes that the interactions between the various cellular components are continuous processes with no uncertainty, while the analysis we present for the bacteriophage λ is carried in a more natural stochastic context. The stochastic phenomena are particularly interesting in multi-stable systems where the interaction of large noise intensities could, for example, lead to random switching from one cellular state to another. Here we demonstrate how our algorithms can be used to alleviate the computational burden involved in computing bounds on probabilities for the occurrence of rare biological events, using a genetic switch implemented in the bacteriophage λ.

3.1.1 Mathematical Background

Robust Stability Analysis At the deterministic level, biological networks are usually modeled as a set of autonomous Ordinary Differential Equations (ODEs) of the form:

$$\dot{x} = f(x), \tag{4}$$

where $\dot{x}$ denotes the derivative of x with respect to time, and $x = (x_1, \ldots, x_n)$ are the state variables. Here f is a function that takes points in $\mathcal{D}$, a subset of $\mathcal{R}^n$ and maps them to $\mathcal{R}^n$, i.e., $f : \mathcal{D} \to \mathcal{R}^n$ with $\mathcal{D} \subseteq \mathcal{R}^n$. We assume further that f is locally Lipschitz for $x \in \mathcal{D}$, a condition that guarantees local existence and uniqueness of solutions [54]. Let x^* be an equilibrium point of (4), i.e. $f(x^*) = \dot{x}^* = 0$. Without loss of generality, we assume that $x^* = 0$ — a simple change of coordinates can achieve this — and study the stability properties of this equilibrium. The following notions of stability is standard.

Definition 1 *Let $\|\cdot\|$ denote a norm in $\mathcal{R}^n$. The zero equilibrium of (4) is:*

- *Stable, if for each $\epsilon > 0$ there is $\delta = \delta(\epsilon) > 0$ such that*

$$\|x(0)\| < \delta \Rightarrow \|x(t)\| < \epsilon, \quad \forall\ t \geq 0.$$

- *Asymptotically stable if it is stable, and δ can be chosen such that*

$$\|x(0)\| < \delta \Rightarrow \lim_{t \to \infty} x(t) = 0.$$

Theese definitions of stability involve $\epsilon - \delta$ formulations. Generally speaking, the stability condition requires that if the initial condition is close to the equilibrium, then the system trajectory will stay close to the equilibrium; the asymptotic stability condition not only asks for stability, but also asymptotic convergence to the equilibrium point. Such conditions give the impression that a complete description of the flow of the vector field is required to answer stability questions. It is fortunate that in many cases this is not essential; instead, stability can be proved directly by exhibiting a so-called *Lyapunov function* [54].

Theorem 2 ([54]) *Consider the system (4), and let $\mathcal{D} \subseteq \mathcal{R}^n$ be a neighborhood of the origin. If there is a continuously differentiable function $V : \mathcal{D} \to \mathcal{R}$ such that the following two conditions are satisfied:*

1. *$V(x) > 0$ for all $x \in \mathcal{D} \setminus \{0\}$ and $V(0) = 0$, i.e., $V(x)$ is positive definite in $\mathcal{D}$;*
2. *$-\dot{V}(x) = -\frac{\partial V}{\partial x} f(x) \geq 0$ for all $x \in \mathcal{D}$, i.e., $\dot{V}(x)$ is negative semidefinite in $\mathcal{D}$;*

then the origin is a stable *equilibrium. If in condition (2) above, $\dot{V}(x)$ is* negative definite *in $\mathcal{D}$, then the origin is* asymptotically stable. *If $\mathcal{D} = \mathcal{R}^n$ and $V(x)$ is radially unbounded, i.e., $V(x) \to \infty$ as $\|x\| \to \infty$, then the result holds globally.*

$V(x)$ in the previous theorem is called a Lyapunov function, while the surface $V(x) = a$ for some $a > 0$ is called a Lyapunov surface. Therefore, the condition $\dot{V}(x) \leq 0$ indicates that when a trajectory of the system crosses $V(x) = a$, it subsequently stays within the set $\{x \in \mathbb{R}^n | V(x) \leq a\}$. Furthermore, the condition $\dot{V}(x) < 0$ indicates that a trajectory moves from one Lyapunov surface to another with a smaller a, eventually approaching the origin.

Constructing Lyapunov functions for nonlinear systems is notoriously difficult. In the past, this led to the development of other methods for assessing nonlinear system properties. For example, in Lyapunov's indirect method, one proceeds by linearizing the vector field about the equilibrium and the stability properties of the original nonlinear system are inferred from the stability properties of the linearized system. However, such stability results are valid only locally and the procedure is inconclusive when the linearized system has imaginary axis eigenvalues. Other methodologies involve absolute stability theory [27], Linear Parameter Varying (LPV) embeddings [36, 73, 105], and Integral Quadratic Constraint (IQC) formulations [67].

Amid these difficulties, a new computationally attractive methodology for nonlinear stability analysis has been recently introduced by Parrilo in [78]. It is based on the sum of squares decomposition (cf. Section 2.1) and uses semidefinite programming to construct functions that satisfy the conditions in Lyapunov's Theorem 2. The methodology generalizes the linear matrix inequality method for constructing quadratic Lyapunov functions for linear time-invariant systems to constructing Lyapunov functions for nonlinear systems. Assume for simplicity that $f(x)$ is a polynomial vector field, i.e., $f_i(x)$ are polynomials in $x = (x_1, \ldots, x_n)$ for $i = 1, \ldots, n$. Suppose that we also wish to construct a $V(x)$ that is also polynomial in x. In this case, the two conditions in Theorem 2 become polynomial nonnegativity conditions. To circumvent the difficult task of testing them directly, we can restrict our attention to the case in which the two conditions admit sum of squares (SOS) decompositions.

In order to impose positive *definiteness* (rather than positive semi-definiteness) on $V(x)$, we construct an auxiliary positive definite 'shaping' function $\varphi(x)$ as follows:

$$\varphi(x) \;=\; \sum_{i=1}^{n} \sum_{j=1}^{d} \epsilon_{ij} x_i^{2j}, \quad \sum_{j=1}^{m} \epsilon_{ij} \geq \gamma \;, \forall \; i = 1, \ldots, n, \tag{5}$$

where $\gamma > 0$, and $\epsilon_{ij} \geq 0 \; \forall \; i, j$. This makes $\varphi(x) > 0$, i.e. positive definite. If we then impose $V(x) - \varphi(x)$ to be a SOS, we get the obvious relation

$$V(x) - \varphi(x) \geq 0 \Rightarrow V(x) \geq \varphi(x) > 0. \tag{6}$$

Overall, we have the following proposition:

Proposition 3 *Given a polynomial $V(x)$ of degree $2d$, let $\varphi(x)$ be given by Equation (5). Then, the condition*

$$V(x) - \varphi(x) \text{ is a sum of squares} \tag{7}$$

guarantees the positive definiteness of $V(x)$.

In light of this proposition, testing global stability for $\mathcal{D} = \mathcal{R}^n$ following the conditions of Theorem 2 can be formulated directly as SOS conditions. It can be specifically formulated as the following sum of squares program:

Program 4 *To construct a Lyapunov function for system (4), find a polynomial $V(x)$, $V(0) = 0$, and a positive definite function $\varphi(x)$ of the form (5), such that*

$$V(x) - \varphi(x) \text{ is SOS} \tag{8}$$

$$-\frac{\partial V}{\partial x} f(x) \text{ is SOS} \tag{9}$$

The function $V(x)$ found this way is a Lyapunov function for system (4) and the zero equilibrium of (4) is stable since the above program guarantees that $V(x)$ is positive definite and also that $\dot{V}(x)$ is negative semidefinite. Note also that by construction $\varphi(x)$ is radially unbounded; therefore, $V(x)$ will also be radially unbounded, and the stability property holds globally [54]. Also, if condition (9) is replaced by

$$-\frac{\partial V}{\partial x} f(x) - \psi(x) \text{ is SOS}, \tag{10}$$

where $\psi(x)$ is a positive definite polynomial constructed as per (5), then $\dot{V}(x)$ is negative definite and the origin is globally asymptotically stable.

To tackle more biologically relevant problems, we extend here Lyapunov's theorem to systems that evolve under equality, inequality, and integral constraints. This is a very general class of systems, special cases of which are differential algebraic equations, robust stability analysis and performance evaluation. Furthermore, this extension allows for the treatment of non-polynomial vector fields.

Inequality constraints arise naturally in biological networks, as the states in biochemical reactions (concentrations of species) are always non-negative. The same type of inequality constraints can be further used to describe *uncertain parameter sets* for the study of robust stability of systems in the presence of parametric uncertainty. Equality constraints also prove useful in robust stability analysis where they appear as constraints guaranteeing that the equilibrium of the system is at the origin. Equality constraints can also naturally appear in systems that are constrained to evolve over a manifold [26]. For example, equality constraints in biological systems can arise due to conservation laws – usually in the form of mass balance equations. Finally, integral type constraints can also be incorporated. In particular, one can consider Integral Quadratic Constraints (IQCs) [67], which provide a rich framework that encapsulates many types of uncertainty and unmodelled dynamics: dynamic, time-varying and L_2 bounded uncertainty, just to name a few. Performance calculations such as L_2 input-output gain estimation can also be formulated using IQCs .

To put these ideas in a more rigorous setting, we consider the nonlinear system

$$\dot{x} \quad = \quad f(x,u), \tag{11}$$

with the following inequality, equality, and integral constraints that are satisfied by x and u:

$$a_{i_1}(x,u) \leq 0, \quad \text{for } i_1 = 1,\ldots,N_1, \tag{12}$$
$$b_{i_2}(x,u) = 0, \quad \text{for } i_2 = 1,\ldots,N_2, \tag{13}$$
$$\textstyle\int_0^T c_{i_3}(x,u)dt \leq 0, \text{for } i_3 = 1,\ldots,N_3, \text{ and } \forall\ T \geq 0. \tag{14}$$

Here $x \in \mathbb{R}^n$ is the state of the system, and $u \in \mathbb{R}^m$ is a collection of auxiliary variables (such as inputs, non-polynomial functions of states, uncertain parameters, etc). We assume that $f(x,u)$, apart from the required Lipschitz conditions for existence of solutions, has no singularity in $\mathcal{D}$, where $\mathcal{D} \subseteq \mathbb{R}^{n+m}$ is defined as

$$\mathcal{D} = \{(x,u) \in \mathbb{R}^{n+m} \mid a_{i_1}(x,u) \leq 0, b_{i_2}(x,u) = 0, \forall i_1 \text{ and } i_2\}.$$

Without loss of generality, it is also assumed that $f(x,u) = 0$ for $x = 0$ and $u \in \mathcal{D}_u^0$, where $\mathcal{D}_u^0 = \{u \in \mathbb{R}^m | (0,u) \in \mathcal{D}\}$. The following theorem is an extension of Lyapunov's stability theorem, and can be used to prove that the origin is a stable equilibrium of the above system. It uses a technique reminiscent of the well-known S-procedure [123] in nonlinear and robust control theory [17], which is similar to adjoining the equality, inequality and IQC conditions to the Lyapunov conditions using appropriate multipliers.

Theorem 5 *[76] Suppose that for system (11), there exist functions $V(x)$, $p_{1_{i_1}}(x,u) \geq 0$, $p_{2_{i_1}}(x,u) \geq 0$, $q_{1_{i_2}}(x,u)$, $q_{2_{i_2}}(x,u)$ and constants $r_{i_3} \geq 0$ such that*

$$V(x) + \sum p_{1_{i_1}}(x,u)a_{i_1}(x,u) + \sum q_{1_{i_2}}(x,u)b_{i_2}(x,u) > 0, \tag{15}$$
$$-\frac{\partial V}{\partial x}f(x,u) + \sum p_{2_{i_1}}(x,u)a_{i_1}(x,u) + \sum q_{2_{i_2}}(x,u)b_{i_2}(x,u) + \sum r_{i_3}c_{i_3}(x,u) \geq 0 \tag{16}$$

Then the origin of the state space is a stable equilibrium of the system.

When the vector field $f(x,u)$ is rational, i.e., $f(x,u) = \frac{n(x,u)}{d(x,u)}$ with $d(x,u) \neq 0$ in $\mathcal{D}$, condition (16) can be multiplied by the non-vanishing denominator. Rational vector fields are common in biological systems, usually arising from Michaelis-Menten approximations.

Theorem 5 can now be used to analyze various cases of systems with constraints. Now, we demonstrate how $V(x)$ in Theorem 5 can be constructed using the SOS technique. For this, we need to make some assumptions.

- The vector field $f_x(x,u)$ is assumed to be polynomial or rational, and the constraint functions $a_{i_1}(x,u)$, $b_{i_2}(x,u)$, $c_{i_3}(x,u)$ are assumed to be polynomial. This assumption may be removed through a recasting process [77].
- We search for a bounded degree *polynomial* Lyapunov function V and multipliers p_{i_1}, q_{i_1}, $i_1 = 1,\ldots,N_1$ and p_{i_2}, q_{i_2}, $i_2 = 1,\ldots,N_2$.

Under these assumptions, the search for a Lyapunov function can be relaxed by the following proposition in which all '$\geq$' conditions are essentially relaxed to SOS conditions, making their search computationally efficient using semidefinite programming and SOSTOOLS.

Proposition 6 *Suppose that for system (11) with* $f(x,u) = \frac{n(x,u)}{d(x,u)}$ *where* $n(x,u)$ *and* $d(x,u)$ *are polynomials and* $d(x,u) > 0$ *in* $\mathcal{D}$*, there exist polynomial functions* $V(x)$*,* $p_{1_{i_1}}(x,u)$*,* $p_{2_{i_1}}(x,u)$*,* $q_{1_{i_2}}(x,u)$*,* $q_{2_{i_2}}(x,u)$*, a positive definite function* $\varphi(x)$ *of the form given in Equation 5 and constants* $r_{i_3} \geq 0$ *such that*

$$V(x) + \sum p_{1_{i_1}}(x,u)a_{i_1}(x,u) + \sum q_{1_{i_2}}(x,u)b_{i_2}(x,u) - \varphi(x) \text{ is SOS,}$$

$$p_{1_{i_1}}(x,u), p_{2_{i_1}}(x,u) \text{ are SOS for } i_1 = 1,\ldots,N_1, \tag{17}$$

$$d(x,u)\begin{pmatrix} -\frac{\partial V}{\partial x}f(x,u) + \sum p_{2_{i_1}}(x,u)a_{i_1}(x,u) \\ +\sum q_{2_{i_2}}(x,u)b_{i_2}(x,u) + \sum r_{i_3}c_{i_3}(x,u) \end{pmatrix} \text{ is SOS.} \tag{18}$$

Then the origin of the state space is a stable equilibrium of the system.

It is now clear how $V(x)$, $p_{1_{i_1}}(x,u)$, $p_{2_{i_1}}(x,u)$, $q_{1_{i_2}}(x,u)$, $q_{2_{i_2}}(x,u)$, the constants r_{i_3} and the positive definite function $\varphi(x)$ can be constructed using SOSTOOLS [91], and a program similar to Program 4 can be constructed.

Model Invalidation Model validation provides a way to evaluate the ability of a proposed model to represent observed system behaviors. However, as often pointed out in the literature, "model validation" is actually a misnomer [30, 82, 108]. It is impossible to validate a model, because to do so requires an infinite number of experiments and data. The role of model validation techniques is rather to *invalidate* a model, by proving that some experimental data are inconsistent with the model, thus indicating that a refinement of the model is required.

We will present a methodology recently developed for invalidation of continuous-time nonlinear models with uncertain parameters [84]. The methodology is based on functions of state-parameter-time termed *barrier certificates*. The existence of a barrier certificate generates a contradiction between model and some time-domain experimental data, in the sense that some level sets of this certificate act as barriers between possible model trajectories and data. With this methodology, model validation of a very large class of continuous-time models, including differential-algebraic models [19], models with uncertain inputs [67], models with memoryless and dynamic uncertainties [54, 67], hybrid models [119], and their combinations. Moreover, similar to Lyapunov functions, barrier certificates can be computed using the sum of squares decomposition and semidefinite programming, e.g., using the SOSTOOLS software [91, 115].

In the simplest setting, consider again the system of ordinary differential equations

$$\dot{x}(t) = f(x(t), p, t), \tag{19}$$

where $x(t) \in \mathbb{R}^n$ is the vector of state variables, t is time, and $p \in \mathbb{R}^m$ is the parameter vector, assumed to take its value in a set $P \subset \mathbb{R}^m$. Let an experiment be performed with the real system, and two measurements be taken at time $t = 0$ and $t = T$. Suppose that these measurements indicate that $x(0) \in X_0$ and $x(T) \in X_T$, where both X_0 and X_T are subsets of $\mathbb{R}^n$. In addition, assume that $x(t) \in X$ for all $t \in [0,T]$, where $X \subseteq \mathbb{R}^n$. The invalidation problem can then be stated as follows:

Problem 7 *Given the model (19), parameter set* P*, and trajectory information* $\{X_0, X_T, X\}$*, prove that for all possible parameter* $p \in P$*, the model (19) cannot produce a trajectory* $x(t)$ *such that* $x(0) \in X_0$*,* $x(T) \in X_T$*, and* $x(t) \in X, \forall t \in [0,T]$*.*

If such a proof in Problem 7 can be found, then we say that the model (19) and parameter set P are invalidated by $\{X_0, X_T, X\}$.

Traditional approaches for solving this problem include exhaustive simulation of (19) using parameters p and initial conditions $x(0)$ sampled randomly from P and X_0. If after many such simulations no trajectory $x(t)$ that satisfies the initial hypothesis can be found, then an inconsistency is concluded. Indeed simulation (possibly after parameter fitting) is a good way for proving that a model can reproduce *some* behaviors of the system it represents. However, for proving inconsistency, the required number of simulation runs will soon become prohibitive. Moreover, a proof by simulation alone is *never exact*, simply because it is impossible to test all p and $x(0)$.

Instead of exhaustive simulations, our method relies on the existence of a function of state-parameter-time, which we term barrier certificate. A barrier certificate gives an *exact* proof of inconsistency by providing a barrier between possible trajectories of the model starting at X_0 and the final measurement X_T. This is accomplished without performing any simulation nor computing the flow of the model. The method is summarized in the following theorem.

Theorem 8 ([84]) *Let the model (19) and the sets P, X_0, X_T, X be given, with $f(x,p,t)$ being continuous in x and t. Suppose that there exists a real-valued function $B(x,p,t)$ that is differentiable with respect to x and t, such that*

$$B(x_T,p,T) - B(x_0,p,0) > 0 \qquad \forall (x_T,x_0,p) \in X_T \times X_0 \times P, \tag{20}$$

$$\frac{\partial B}{\partial x}(x,p,t) f(x,p,t) + \frac{\partial B}{\partial t}(x,p,t) \leq 0 \qquad \forall (x,p,t) \in X \times P \times [0,T]. \tag{21}$$

Then the model (19) and its associated parameter set P are invalidated by $\{X_0, X_T, T\}$. (In the sequel, we will call the function $B(x,p,t)$ a barrier certificate.)

Similar to the case of Lyapunov functions, construction of barrier certificates is in general not easy. However, for models with polynomial vector fields and sets P, X_{T_i}, X, U_i described by polynomial equalities and inequalities, a tractable computational relaxation for constructing barrier certificates exists. The relaxation is provided by the sum of squares decomposition and semidefinite programming, much in the spirit of the algorithmic construction of Lyapunov functions for nonlinear systems presented in the previous subsection.

Stochastic Verification Computing bounds on probability in inherently stochastic biological systems, as well as the statistics for the occurrence or lack thereof of decisive or catastrophic events is a crucial problem. However, such computations are usually very challenging, and can be carried analytically or numerically only for a handful of examples. In this section, a technique based on the sum of squares approach that was recently developed [87, 88] for the algorithmic computation of probability bounds on the occurrence of these events will be described.

For our exposition, we consider a complete probability space $(\Omega, \mathcal{F}, P)$ and a standard $\mathbb{R}^m$-valued Wiener process w defined on this space [72]. The class of models that we use is stochastic differential equations of the form

$$dx(t) = f(x(t))dt + g(x(t))dw(t) \tag{22}$$

where $x(t) \in \mathbb{R}^n$, and $t \geq 0$. We denote the state space, the set of initial states, and the set of decisive/unsafe states by $\mathcal{X}, \mathcal{X}_0$, and $\mathcal{X}_u$, respectively. All of these are subsets of $\mathbb{R}^n$ and assumed

compact. To guarantee the existence and uniqueness of solution, we will also assume that both $f(x)$ and $g(x)$ satisfy the local Lipschitz condition and the linear growth condition on $\mathcal{X}$. For bounded $\mathcal{X}$, the last condition can be replaced by the boundedness of f and g on $\mathcal{X}$. It can then be shown that the process $x(t)$ described above is a strong Markov process.

Since in general the process $x(t)$ is not guaranteed to always lie inside the set $\mathcal{X}$, we define the stopped process $\tilde{x}(t)$ corresponding to $x(t)$ and $\mathcal{X}$ as follows.

Definition 9 *Suppose that τ is the first time of exit of $x(t)$ from the open set Int($\mathcal{X}$). The stopped process $\tilde{x}(t)$ is defined by*

$$\tilde{x}(t) = \begin{cases} x(t) & \text{for } t < \tau, \\ x(\tau) & \text{for } t \geq \tau. \end{cases}$$

The stopped process $\tilde{x}(t)$ satisfies various properties. For example, it inherits the right continuity and strong Markovian property of $x(t)$. Furthermore, in most cases the so-called infinitesimal generator corresponding to $\tilde{x}(t)$ is identical to the one corresponding to $x(t)$ on the set $\mathrm{Int}(\mathcal{X})$, and is equal to zero outside of this set [58]. This will be implicitly assumed in our analysis.

Having defined the system and the stopped process $\tilde{x}(t)$, we can now state the stochastic verification problem as follows.

Problem 10 *Given the system (22) and the sets $\mathcal{X}$, $\mathcal{X}_0$ and $\mathcal{X}_u$, compute an upper bound for the probability of a process $\tilde{x}(t)$ starting at $\mathcal{X}_0$ to reach $\mathcal{X}_u$. In other words, find $\gamma \in [0, 1]$ such that $P\{\tilde{x}(t) \in \mathcal{X}_u \text{ for some } t \geq 0 \mid \tilde{x}(0) = x_0\} \leq \gamma$ for all $x_0 \in \mathcal{X}_0$.*

Obviously, it is of interest to obtain an upper bound γ that is as tight as possible. Our approach to solve the above problem is based on finding an appropriate function $B(x)$ from which we can deduce an upper bound γ. We will also call the function $B(x)$ a barrier certificate, as it will need to satisfy some conditions that can be considered as the stochastic counterpart of the conditions in the previous subsection. For example, instead of requiring the value of $B(x(t))$ to be non-increasing along time, we ask that the *expected value* of $B(x(t))$ decreases or stays constant as time increases. A process satisfying such a property is called a supermartingale (see [72] for the technical definition). Using a known supermartingale inequality [58], the probability bound can then be inferred from $B(x)$, as summarized in the following theorem.

Theorem 11 ([88]) *Let the stochastic differential equation (22) and the sets $\mathcal{X}, \mathcal{X}_0, \mathcal{X}_u$ be given, and consider the stopped process $\tilde{x}(t)$. Suppose that there exists a twice continuously differentiable function $B : \mathbb{R}^n \to \mathbb{R}$, such that*

$$B(x) \geq 0 \ \forall x \in \mathcal{X} \tag{23}$$

$$B(x) \geq 1 \ \forall x \in \mathcal{X}_u \tag{24}$$

$$B(x) \leq \gamma \ \forall x \in \mathcal{X}_0 \tag{25}$$

$$\frac{\partial B}{\partial x}(x) f(x) + \frac{1}{2}\mathrm{Trace}\left(g^T(x) \frac{\partial^2 B}{\partial x^2}(x) g(x) \right) \leq 0 \ \forall x \in \mathcal{X} \tag{26}$$

then

$$P\{\tilde{x}(t) \in \mathcal{X}_u \text{ for some } t \geq 0 \mid \tilde{x}(0) = x_0\} \leq \gamma \tag{27}$$

for all $x_0 \in \mathcal{X}_0$.

With regard to computation, an upper bound γ and a barrier certificate $B(x)$ which certifies the upper bound can be computed by formulating conditions (23)–(26) as a sum of squares optimization problem. Furthermore, γ can be chosen as the objective function whose value is to be minimized. The minimum value of γ obtained from the optimization will be the tightest upper bound for a given set of candidate barrier certificates. Obviously, as we include more candidate barrier certificates to this set, we may get a better bound, although there is a trade-off between using a larger set and the computational complexity of finding a barrier certificate within the set.

3.1.2 Application to Gene Regulatory Networks

In order to put the biological models of this section in context, we present a brief introduction to the cellular processes involved in gene expression, in addition to the concept of a gene regulatory network.

Gene Regulatory Networks and the Central Dogma of Molecular Biology The synthesis of cellular proteins is a multi-step process that involves the use of various cellular machines [1]. One very important machine in bacteria is the so called RNA polymerase (RNAP). RNAP is an enzyme that can be recruited to transcribe any given gene. However, RNAP bound to regulatory sigma factors recognizes specific sequences in the DNA, referred to as the *promoter*. Whereas the role of RNAP is to transcribe genes, the main role of σ factors is to recognize the promoter sequence and signal to RNAP in order to initiate the transcription of the appropriate genes. The transcription process itself consists of synthesizing a messenger RNA ($mRNA$) molecule that carries the information encoded by the gene. Here, RNAP acts as a "reading head" transcribing DNA sequences into mRNA. Once a few nucleotides on the DNA have been transcribed, the σ-factor molecule dissociates from RNAP, while RNAP continues transcribing the genes until it recognizes a particular sequence called a *terminator sequence*. At this point, the mRNA is complete and RNAP disengages from the DNA. During the transcription process, ribosomes bind to the nascent $mRNA$ and initiate translation of the message. The process of translation consists of sequentially assembling amino acids in an order that corresponds to the mRNA sequence, with each set of three nucleotides corresponding to a single unique amino acid. This combined process of gene transcription and mRNA translation constitutes gene expression, and is often referred to as the *central dogma* of molecular biology.

Gene regulatory networks can be broadly defined as groups of genes that are activated or deactivated by particular signals and stimuli, and as such produce or halt the production of certain proteins. Through combinatorial logic at the gene or the end-product protein level, these networks orchestrate their operation to regulate certain biological functions such as metabolism, development, or the cellular clocks. Regulation schemes in gene regulatory networks often involve positive and negative feedback loops. A simple scheme consists, for example, of a protein that binds to the promoter of its own gene and shields it from the RNAP-σ complex, thereby auto-regulating its own production. When interfaced and connected together according to a certain logic, a network of such building blocks (and possibly others possessing different architectures and components) generates intricate systems that possess a wide range of dynamical behaviors and functionalities.

Robust Stability Analysis of the Heat Shock Response in *E. coli* In this section, we describe the biology of the heat shock response system, an essential cellular mechanism conserved in most organisms. Extensive experimental probing of the architecture of the heat shock system has unraveled a number of feedback and feedforward loops whose presence posed many essential

questions. First, and perhaps foremost, does the current knowledge span all the structural components and interactions in the system or did experimental investigation miss crucial parts? If so, can a model describing the known components identify this knowledge gap when contrasted with data? Even if such investigations confirm the completeness of our knowledge of the system, can we decipher the salient features of its architecture and identify the functional roles of its parts. For example, what is the functional relevance of the seemingly redundant loops in the heat shock system? Is the system robust to operational fluctuations? If so, how is this robustness connected to the architecture?

These questions are of course of general relevance to many biological systems, but are at the same time immensely challenging. Here, we use the heat shock response as a case study to formulate and address concisely particular aspects of robustness analysis and model invalidation, all in the context of our new mathematical machinery. The heat shock response is particularly suited for such an investigation because of the wealth of experimental and mathematical information available for the system [32].

The Heat Shock Response in *E. coli* High temperatures cause cell proteins to unfold from their normal shapes, resulting in malfunctioning and eventually death of the cell. Cells have evolved gene regulatory mechanisms to counter the effects of heat shock by expressing specific genes that encode heat shock proteins (hsps) whose role is to help the cell survive the consequence of the shock. In *E. coli*, the heat shock (HS) response is implemented through an intricate architecture of feedback loops centered around the σ-factor that regulates the transcription of the HS proteins under normal and stress conditions. The enzyme RNA polymerase (RNAP) bound to this regulatory sigma factor, σ^{32}, recognizes the HS gene promoters and transcribes specific HS genes. The HS genes encode predominantly molecular chaperones (DnaK, DnaJ, GroEL, GrpE, etc.) that are involved in refolding denatured proteins and proteases (Lon, FtsH, etc.) that function to degrade unfolded proteins. At physiological temperatures (30°C to 37°C), there is very little σ^{32} present and hence little transcription of the HS genes. When bacteria are exposed to high temperatures, σ^{32} first rapidly accumulates, allowing increased transcription of the HS genes and then declines to a new steady state level characteristic of the new growth temperature. There are two mechanisms by which σ^{32} levels are increased when the temperature is raised [113]. First, the translation rate of the rpoH mRNA (encoding σ^{32}) increases immediately, resulting in a fast 10-fold increase in the concentration of σ^{32} [70]. This mechanism implements what we refer to as the *feedforward control loop*. Second, during steady state growth, σ^{32} is rapidly degraded ($t_{1/2} = 1$ minute), but is stabilized for the first five minutes after temperature upshift, so that its concentration rapidly increases. *In vivo* evidence is consistent with the following titration model for the HS response. DnaK and its cochaperone DnaJ are required for the rapid degradation of σ^{32} by the HS protease FtsH. Raising the temperature produces an increase in the cellular levels of unfolded proteins that then titrate DnaK/J away from σ^{32}, allowing it to bind to RNA polymerase (resulting in increased trancription) and stabilizing it in the process. Together, increased translation and stabilization lead to a transient 15-20 fold increase in the amount of σ^{32} at the peak of the HS response. The accumulation of high levels of HS proteins leads to the efficient refolding of the denatured proteins thereby decreasing the pool of unfolded protein, freeing up DnaK/J to sequester this protein from RNA polymerase. This implements what is referred to as a *sequestration feedback loop*. Furthermore, this sequestration itself promotes the degradation of σ^{32} and results in feedback regulated degradation, mainly by the protease FtsH. We refer to this as the *FtsH degradation feedback loop*. The overall result is a decrease in the concentration of σ^{32} to a new steady state concentration that is dictated by the balance between the temperature-dependent translation of the rpoH mRNA and the level of σ^{32}

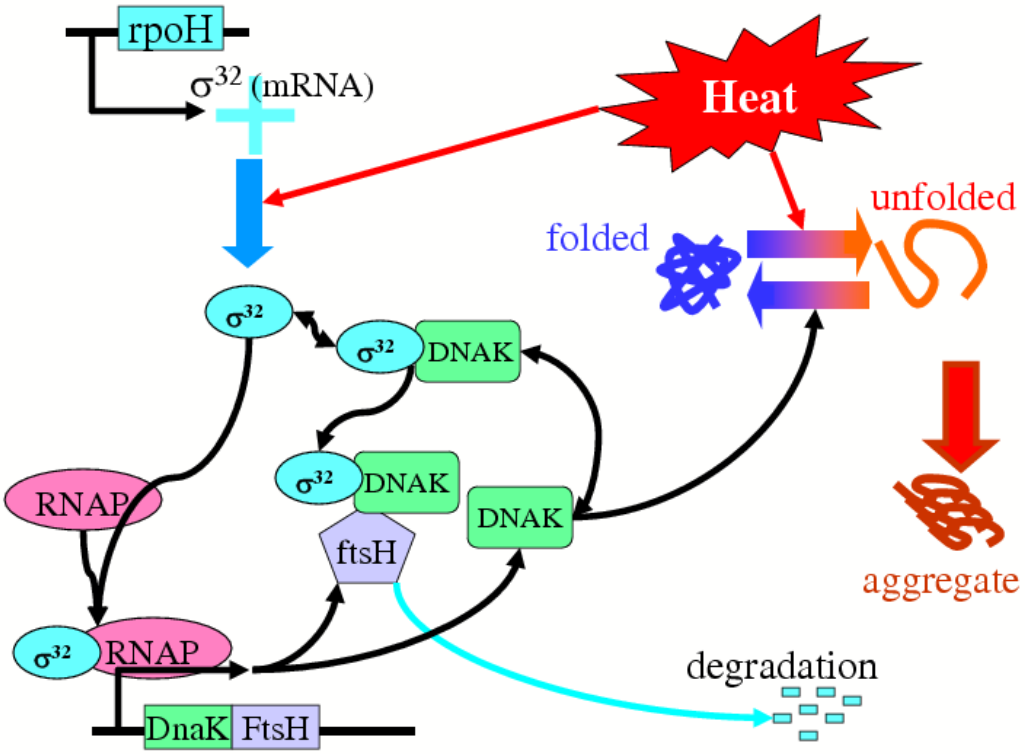

Figure 1: Molecular implementation of the heat shock response system

activity modulated by the hsp chaperones and proteases acting in a negative feedback fashion. The molecular interactions describing the heat shock response are pictorially illustrated in Figure 1.

A Reduced Order Model for the HS Response In a previous study, we have developed a detailed deterministic mathematical model for the heat stress response in *E. coli* [31,32,57]. Specifically, the dynamics described above were modeled using differential rate equations, generating a set of 31 Differential-Algebraic Equations (*DAEs*) of the form

$$\begin{aligned} \dot{X}(t) &= F(t;X;Y) \\ 0 &= G(t;X;Y) \end{aligned}$$

where X is an 11-dimensional vector whose elements are the differential variables and Y is a 20-dimensional vector whose elements are algebraic variables. This form is known as a *semi-explicit* DAE. The model possesses 27 kinetic rate parameters. Subsequently, a reduced order model was derived using insight into the system's architecture and separation principles in time and concentrations. As in the full model, this reduced model involves the dynamics of the basic building blocks of the HS response, namely the σ factor (S), the chaperones (D), and the protein

Parameter	Value
K_d	3 min^{-1}
α_d	0.015 min^{-1}
$\eta(T)$	10 $molecule.min^{-1}$@ T_1 & 60 @ T_2
α_0	0.03 min^{-1}
α_s	3 min^{-1}
K_s	0.05 $molecule^{-1}$
K_u	0.0254 $molecule^{-1}$
$K(T)$	40 min^{-1} @ T_1 & 80 min^{-1} @ T_2
K_{fold}	6000 min^{-1}
P_t	2×10^6 molecules

Table 2: Parameter Values for Heat Shock model

folding mechanism. The model equations are as follows

$$\begin{aligned}
\frac{dD_t}{dt} &= K_d S_f - \alpha_d D_t \\
\frac{dS_t}{dt} &= \eta(T) - \alpha_0 S_t - \alpha_s S : D \\
\frac{dU_f}{dt} &= K(T) P_{folded} - K_{fold} U : D \\
S : D &= K_s . S_f . D_f \\
U : D &= K_u . U_f . D_f \\
D_t &= D_f + U : D + S : D \\
S_t &= S_f + S : D \\
P_t &= P_{folded} + U_f + U : D
\end{aligned} \tag{28}$$

where $U : D$ is the complex formed by the binding of the unfolded proteins U_f to D, $S : D$ is the complex formed by the binding of S to D, and P_t is the total number of proteins in the cell, considered here to be constant. The parameters used in this model are given in Table 2. We replace the algebraic constraints into the initial system (28), then use the facts that $S_t \ll D_t$ and that $U_f \gg 1$ in the wild type bacterial HS response and simplify the expression for S_f and D_f. Simple algebraic manipulations yield a compact description for the reduced order HS model:

$$\begin{aligned}
\frac{dD_t}{dt} &= f_1(D_t, U_f, S_t) - \alpha_d D_t \\
\frac{dS_t}{dt} &= \eta(T) - \alpha_0 . S_t - f_2(D_t, U_f, S_t) \\
\frac{dU_f}{dt} &= K(T)[P_t - U_f] - [K(T) + K_{fold}] D_t
\end{aligned} \tag{29}$$

As in the original equations, the feedforward control is achieved by the temperature dependent function $\eta(T)$ in the ODE describing the dynamics of S_t. $f_1(D_t, U_f, S_t) = K_d \frac{S_t}{1 + \frac{K_s D_t}{1 + K_u U_f}}$ and $f_2(D_t, U_f, S_t) = \alpha_s \frac{\frac{K_s D_t}{1 + K_u U_f}}{1 + \frac{K_s D_t}{1 + K_u U_f}} S_t$ describe the various feedback strategies implemented in the HS response. f_1 is the effect of the sequestration of S by D on D formation, while f_2 reflects the effect of the regulated degradation of S through the action of the sequestration itself. The dynamics of

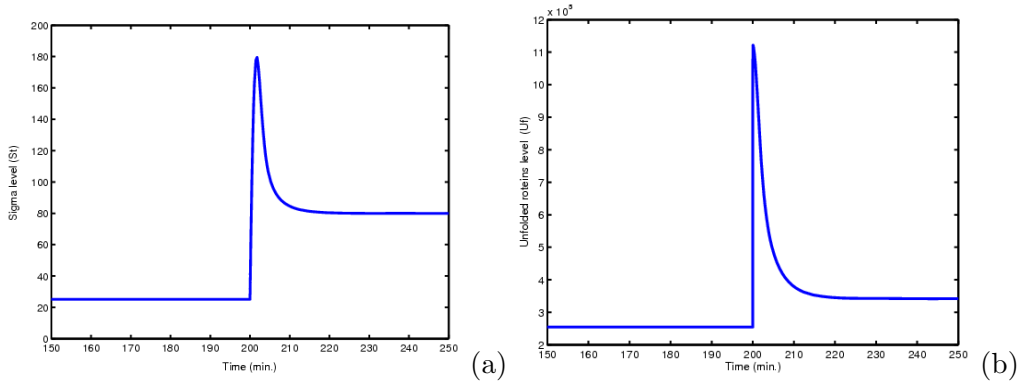

Figure 2: Plots of σ^{32} (a) and unfolded proteins (b) in the heat shock response system

the third state U_f are much faster than those of S_t and D_t. Such stiffness is also strongly present in the full model and creates ill-conditioning and algorithms that do not exploit stiffness are almost certainly doomed to suffer from it. However, stiffness can also be exploited to robustly produce simplified models by singular perturbation, as was done in deriving the 3-state from the full model. By further setting $\frac{dU_f}{dt} = 0$ to obtain a quasi-steady state approximation, the third equation is then replaced by an algebraic one, and the result is again a differential-algebraic equation (DAE). The validity of this approximation has been verified by simulation which showed virtually no difference in the solution of the ODE as compared to that of the DAE. A time course for σ^{32} and the level of unfolded proteins is shown in Figure 2.

Results for Robustness Analysis of the HS System For the heat shock model, we can consider the problem of proving robust stability for the system under parametric uncertainty. We proceed by non-dimensionalizing the states of (29) by their equilibrium values $(D_{t_0}, S_{t_0}, U_{f_0})$, followed by a shifting of the equilibrium of the system to the origin. We then obtain a system with states (x_1, x_2, x_3) that is better conditioned, in the sense that the states are of the same order of magnitude:

$$\begin{aligned} \frac{dx_1}{dt} &= \tilde{f}_1(x_1, x_2, x_3) - \alpha_d x_1 \\ \frac{dx_2}{dt} &= \tilde{\eta}(T) - \alpha_0 x_2 - \tilde{f}_2(x_1, x_2, x_3) \\ \frac{dx_3}{dt} &= K(T)[\tilde{P}_t - x_3] - K_{Tot} x_1 \end{aligned} \tag{30}$$

with $\tilde{f}_1(x_1, x_2, x_3) = \tilde{K}_d \frac{x_2}{1+\frac{\tilde{K}_s x_1}{1+\tilde{K}_u x_3}}$ and $\tilde{f}_2(x_1, x_2, x_3) = \alpha_s x_2 \frac{\frac{\tilde{K}_s x_1}{1+\tilde{K}_u x_3}}{1+\frac{\tilde{K}_s x_1}{1+\tilde{K}_u x_3}}$, and where $\tilde{K}_d = K_d S_{t_0}/D_{t_0}$, $\tilde{K}_s = K_s D_{t_0}$, $\tilde{K}_u = K_u U_{f_0}$, $\tilde{\eta} = \eta/S_{t_0}$, $\tilde{P}_t = P_t/U_{f_0}$ and $K_{Tot} = D_{t_0}(K(T) + K_{fold})/U_{f_0}$. We then use $\frac{dx_3}{dt} = 0$ to get a 2-D state-space (x_1, x_2). To proceed, we first define the region $\mathcal{D}$ in the state-space where a Lyapunov function is to be constructed:

$$\mathcal{D} = \{x_i \in \mathcal{R} : (x_i - x_{i_0})^2 - \gamma_i^2 \leq 0, i = 1, 2\} \tag{31}$$

with $\gamma_i = 0.2$, x_{i_0} denoting the equilibrium of the i-th state. For robust stability analysis purposes, we pick two important parameters, $\tilde{\eta}$ and α_s. $\tilde{\eta}$ depicts the feedforward gain, while α_s forms part

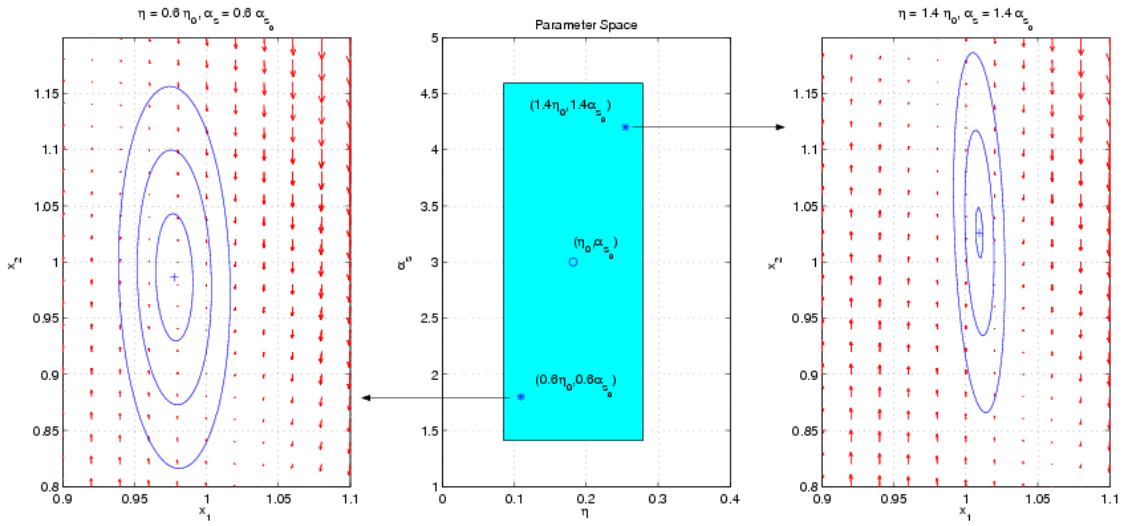

Figure 3: Robust stability of the model of Heat Shock in *E-coli*.

of the feedback gain. We ask whether the equilibrium of the system described by (29) is stable for all values of $\tilde{\eta}$ and α_s in a certain range for $T = T_1$:

$$P = \{\tilde{\eta}, \alpha_s \in \mathcal{R}^2 : (\tilde{\eta} - \tilde{\eta}_0)^2 - (\gamma_3 \tilde{\eta}_0)^2 \leq 0, (\alpha_s - \alpha_{s_0})^2 - (\gamma_4 \alpha_{s_0})^2 \leq 0\},$$

with γ_3 and γ_4 measuring the percentage variation. As these parameters change, the equilibrium of the system also changes. Therefore, in order to fix the equilibrium at the origin, we impose two equality constraints of the form: $\alpha_d \alpha_s \tilde{K}_s x_{1_0}^2 - [\tilde{K}_d(\tilde{\eta}(T_1) - \alpha_0 x_{2_0}) \left(1 + \tilde{K}_u \left(\tilde{P}_t - \frac{K_{\text{Tot}}}{K(T_1)} x_{1_0}\right)\right)] = 0$ and $\tilde{K}_s \tilde{K}_d(\tilde{\eta}(T_1) - \alpha_0 x_{2_0}) - \alpha_s \tilde{K}_s(\tilde{K}_d x_{2_0} - \alpha_d x_{1_0}) = 0$.

Note that the vector field is rational, but this case can be treated using Theorem 5 by multiplying out by the (non-vanishing) common denominator of the vector field. Robust stability analysis is then carried out by constructing a parameter-dependent Lyapunov function, using the results in proposition 6 and SOSTOOLS. We start with a quadratic Lyapunov function V that is not parameterized by any parameters; in this case, we could prove stability for $\gamma_3 = \gamma_4 = 0.45$. When the Lyapunov function is parameterized by α_s and $\tilde{\eta}$, we could construct a Lyapunov function for $\gamma_3 = \gamma_4 = 0.53$. By increasing the complexity of the certificate, we could construct a Lyapunov function for a larger parameter range. In this case, while the equilibrium is stable for even larger parameter sets, the other equilibrium in the system (which is unstable) approaches the equilibrium of interest. Therefore, to prove stability for a larger parameter set, we need to reduce the size of the region D and increase the order of the Lyapunov function. Figure 3 shows the level curves of the Lyapunov function for two sets of parameters in a parameter set with $\gamma_3 = \gamma_4 = 0.53$.

Validation/Invalidation of the Model of Heat Shock Response in E. coli The new methodology in conjunction with SOSTOOLS can be used to address the critical issue of model validation/invalidation in biological modeling. The key ideas of this methodology can be illustrated in the context of the heat-shock example, where at least two feedback loops are involved in the regulation scheme. We will show rigorously that each loop adds its own important function to the overall system and that both are necessary to explain the phenotypic behavior of the heat shock system. In previous work, we have used sensitivity analysis and confirmed that these feedback loops indeed increase the robustness to parametric uncertainty [31]. However, upon disabling

the degradation (FtsH) feedback loop, one observes in simulation that the transient response to a temperature increase becomes considerably slower. Achieving a faster transient response in the absence of this (FtsH) feedback loop necessitates a substantial increase in the protein synthesis rate, and therefore, produces a larger number of chaperones. Therefore, it is reasonable to conjecture that the (FtsH) feedback loop is instrumental in achieving a fast response to the heat disturbance while using a relatively modest number of chaperones.

To actually prove such a conjecture using the invalidation scheme in Section 3.1.1, we will generate some "data" using the model with the degradation (FtsH) loop (29), and compare it to a hypothesized model lacking this feedback. If we denote the state variables (D_t, S_t, U_f) by (x_1, x_2, x_3), then the hypothesized model will just be $\dot{x} = f(x, p)$, where the vector field are defined by (29), without the degradation loop. The parameters p will be defined below. A numerical experiment (i.e., a simulation) with the full system is performed, with the parameters fixed at the nominal values. We observe that the corresponding system trajectory satisfies $x(0) \in X_0$ and $x(25) \in X_T$, where

$$X_0 = \{x \in \mathcal{R}^3 : x_1 \in [0.9D_0, 1.5D_0], x_2 \in [0.9S_0, 1.5S_0], x_3 \in [2.9U_0, 3.1U_0]\} \tag{32}$$

$$X_T = \{x \in \mathcal{R}^3 : x_1 \in [1.5D_0, 2.5D_0], x_2 \in [2S_0, 3S_0], x_3 \in [0.5U_0, 1.5U_0]\} \tag{33}$$

with D_0, S_0, and U_0 denoting their steady state values at low temperature. Note that we use intervals here to take into account the effects of measurement uncertainty, variation of initial conditions, and so on. In addition, we also observe that between time $t = 0$ and $t = 25$, the state variables satisfy $x(t) \in X$, with

$$X = \{x \in \mathcal{R}^3 : x_1 \in [0.9D_0, 2.5D_0], x_2 \in [0.9S_0, 8S_0], x_3 \in [0.2U_0, 4U_0]\}. \tag{34}$$

For the hypothesized model, we will focus on three parameters $p = (K_d, \alpha_0, \eta(T))$, and assume that the rest are fixed at the nominal values. Plausible ranges for these parameters define the parameter set P:

$$\begin{aligned} P = \{(K_d, \alpha_0, \eta(T)) \in \mathcal{R}^3 : 0.5\overline{K}_d \le K_d \le 5\overline{K}_d, \\ 0.5\overline{\alpha}_0 \le \alpha_0 \le 1.5\overline{\alpha}_0, 0.5\overline{\eta(T)} \le \eta(T) \le 1.5\overline{\eta(T)}\}, \end{aligned} \tag{35}$$

where $\overline{K}_d$, $\overline{\alpha}_0$, and $\overline{\eta(T)}$ denote their nominal values. We deliberately make the upper bound for K_d quite large, since one obvious way for obtaining a fast response is to increase the number of chaperones, corresponding to increasing this parameter. With our method, we can find a barrier certificate for these model and data, in effect proving that the model without the degradation (FtsH) loop and with parameters K_d, α_0, $\eta(T)$ satisfying (35) cannot possibly generate a time response that satisfies (32)–(34). This indicates that an inherent mechanism is missing from this model. When the FtsH mechanism is included, obviously there are values for parameters K_d, α_0, and $\eta(T)$ (for example, we can simply choose the nominal values $\overline{K}_d$, $\overline{\alpha}_0$, and $\overline{\eta(T)}$) such that the model has a time response that satisfies (32)–(34).

Stochastic Reachability Analysis of the Bacteriophage Lambda One of the best studied examples of multistability in genetic systems is the bacteriophage λ system [2, 96]. This system has largely been used as a prototype for the investigation of stochasticity in cellular networks, and an illustration of the mathematical challenges that reside in such investigations. Phages are viral organisms that can either be in the lysogenic (latent) or lytic (active) state. If following its infection of *E. coli*, the λ-phage virus enters the lysogenic pathway, it represses its own developmental

functions and integrates its DNA into the host chromosome. Otherwise, it enters the lytic pathway and is active. The dynamics of the transcriptional network underlying the formation of these states are very complex, and have been thoroughly studied [96, 106]. Instead of describing the full complexity of the λ-phage system, we limit ourselves here to a simplified model of the process that still captures the essence of its exhibited bistability, in addition to the interplay between noise and dynamics in its operation.

The λ-Phage One-Dimensional Model The representative simple model we adopt was first discussed by Hasty et al. [46], and is intended to be a minimal model that captures the bistable nature of the λ system. The model describes the dynamics of the CI protein, the product of the cI gene. CI, acting as a dimer, can regulate its own synthesis by binding to the cI gene promoter region OR_2 and increasing transcription or to gene promoter OR_3 and repressing transcription. It can also bind to a third gene promoter OR_1, which we ignore in this model for simplicity. We assume the fast binding reactions (such as binding and dissociation) to be in equilibrium with respect to the slow reactions (such as protein synthesis and degradation). Under this assumption, and following [46], we let X, X_2, and D denote the repressor, repressor dimer, and DNA promoter site respectively. Then, the equilibrium reactions can be written as

$$\begin{aligned} 2X \overset{K_1}{\rightleftharpoons} X_2, \quad D + X_2 \overset{K_2}{\rightleftharpoons} DX_2, \\ D + X_2 \overset{K_3}{\rightleftharpoons} DX_2^*, \quad DX_2 + X_2 \overset{K_4}{\rightleftharpoons} DX_2X_2 \end{aligned} \tag{36}$$

where the DX_2 and DX_2^* complexes denote binding to the OR_2 or OR_3 sites, respectively, DX_2X_2 denotes binding to both sites, and the K_i are forward equilibrium constants. We further set $K_3 \simeq K_2$ and $K_4 \simeq 5K_2$.

The slow reactions are transcription and degradation. At first, we assume that translation is a fast process and therefore lump its dynamics with those of transcription. Later, we will relax this assumption in order to construct a higher order model of the system. The molecular reactions describing the slow dynamics are given by

$$DX_2 + P \xrightarrow{k_t} DX_2 + P + nX, \quad X \xrightarrow{k_d} A$$

where P denotes the concentration of RNA polymerase, and n is the number of proteins per mRNA transcript. To model the system, we state variables as $x = [X], y = [X_2], d = [D], u = [DX_2], v = [DX_2^*]$, and $z = [DX_2X_2]$, where [.] denotes concentration.

The evolution of the concentration of the repressor x can then be described by

$$\dot{x} = -2k_1x^2 + 2k_{-1}y + nk_tp_0u - k_dx + q \tag{37}$$

Here, the concentration of RNA polymerase p_0 is assumed to remain constant during time. The parameter q is the basal rate of production of CI, i.e., the expression rate of the cI gene in the absence of a transcription factor. Because the reactions in (36) are fast, y, u, and d have algebraic expressions in terms of x. We further use the fact that total amount of DNA promoter sites is constant at a value d_T to derive an expression of d in terms of x. The resulting one-dimensional model for x is then given by

$$\dot{x} = \frac{nk_tp_0d_tK_1K_2x^2}{1 + 2K_1K_2x^2 + 5K_1^2K_2^2x^4} - k_dx + q \tag{38}$$

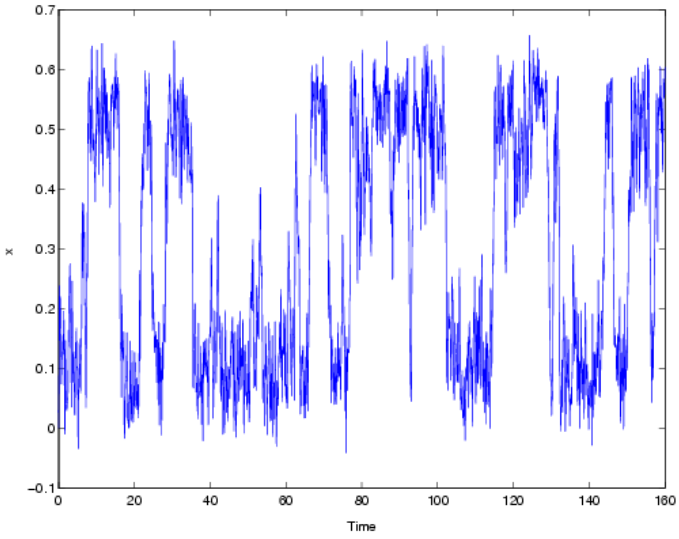

Figure 4: Switching between equilibria in the λ system caused by noise

To reduce the size of the parameter space, we eliminate two of the parameters in (38) by rescaling the repressor concentration x and time. To this end, we define the dimensionless variables $\tilde{x} = x\sqrt{K_1 K_2}$ and $\tilde{t} = t(q\sqrt{K_1 K_2})$. On substitution into (38) we obtain

$$\dot{x} = \frac{\alpha x^2}{1 + 2x^2 + 5x^4} - \gamma x + 1 \tag{39}$$

where the tildes have been suppressed, and we have defined $\alpha = n k_t p_0 d_T / q$ and $\gamma = k_d/(q\sqrt{K_1 K_2})$.

For this equation, there are two types of behavior, depending on the choice of the parameters α and γ. As the parameters vary, the number of equilibria in the system changes from 1 to 3 and then back to 1. When there is only one stable equilibrium and all concentrations evolve to that equilibrium. When there are 3 equilibria, two of which are stable and the other unstable. Therefore, the asymptotic behavior of the system is dependent on the initial condition. Such a behavior is known us bistable, and appears frequently in many areas of engineering and physics.

Stochastic Analysis of the One-Dimensional SDE Formulation of Phage λ Although a given trajectory of the phage λ system will converge to one of the two equilibria at steady-state, it might leave this steady state and switch to the other one under the action of the biochemical noise affecting the many reactions occurring in the system. We show this behavior in Figure 4 where a realization based on the stochastic simulation algorithm of Gillespie is shown. In this section, we are concerned with the characterization of the statistics of such switching events.

However, here we consider a Langevin SDE formulation of the λ-phage system. To do that, we start by incorporating additive noise in the deterministic rate equation of (39). If we take the dynamical variable x to represent the repressor number within a colony of cells, we can crudely think of an additive white noise term as a randomly varying external field acting on the biochemical reactions, hence accounting for the impact of the environment on the system. The resulting SDE is given by

$$dx(t) = f(x)dt + \sigma d\xi(t) \tag{40}$$

where $f(x)$ is the right-hand side of (39), $\xi(t)$ is a Wiener process, and σ is a scalar. The presence of this noise source poses new questions. For example, while in the deterministic description

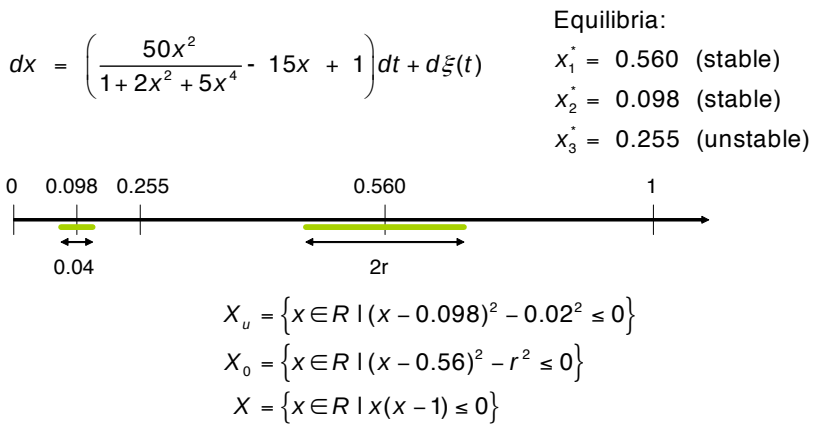

Figure 5: Setup for computing the bound of the probability of false switching because of noise, for system (40).

trajectories that reach either steady state come to rest there, in the stochastic case they never do so because of noise. One can then ask whether there is a large probability of noise-induced switching between equilibria within a given time frame, then describe its correlation with the noise intensity affecting the system. These quantities can be computed through the exact analytical solution of the SDE. However, when such solution cannot be computed explicitly as it is often the case, statistics determined from extensive simulations are usually used. Alternatively here, we propose that the sum of squares and SOSTOOLS machinery can be used as an efficient algorithmic method for such investigations. We specifically illustrate this point by investigating the situation where the model in (39) is started close to one of the two stable equilibria, and an estimate of the the probability of transition to a region around the other stable equilibrium is computed. We will refer to this as *the bound on the probability of false switching caused by noise*. For example, the setup for this problem is shown in Figure 5 where we are interested in whether an initial trajectory from a point inside a set of variable size r centered at the high equilibrium

$$\mathcal{X}_0 = \{x \in \mathbb{R} | (x - 0.56)^2 - r^2 \leq 0\} \tag{41}$$

can ever reach a region around the other equilibrium,

$$\mathcal{X}_u = \{x \in \mathbb{R} | (x - 0.098)^2 - 0.02^2 \leq 0\}. \tag{42}$$

when the total state-space of interest is

$$\mathcal{X} = \{x \in \mathbb{R} | x(x-1) \leq 0\}. \tag{43}$$

This question, however, is ill-posed if the time horizon over which this probability is to be computed is not bounded. Indeed, the way the noise enters in (40) means that if the state-space $\mathcal{X}$ is large enough (say the whole real line) and the time is infinite, the probability of reaching every point in the state-space should be 1.

A more meaningful biological question that our methods can answer is the following. Starting from a region around one equilibrium, estimate the probability of reaching the other equilibrium *in a finite time horizon*, say from $t = 0$ to 2 non-dimensional time units. Time is now another variable in the system, and instead of constructing a time-independent $B(x)$, we construct a time-dependent $B(x, t)$ to estimate this probability. We also increase the state-space adequately, so

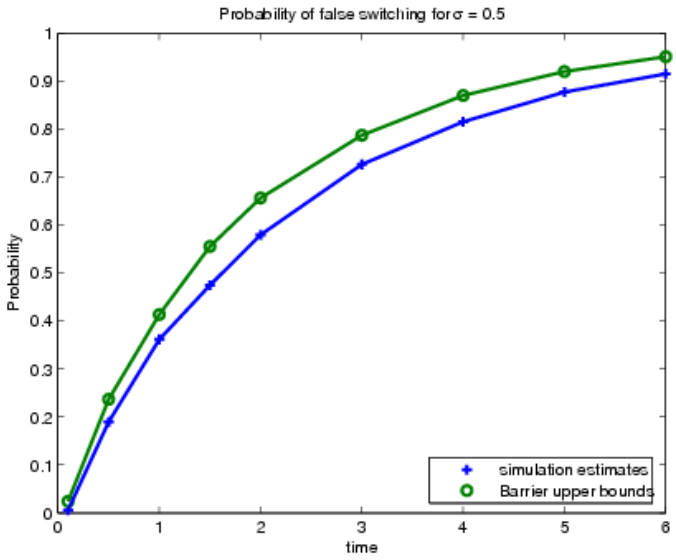

Figure 6: Probability of false switching from the high steady to the low steady state as a function of time under the influence of noise. The blue line depicts probabilities computed through the use of barrier functions while the green line depicts the probabilities computed from 2000 realizations obtained from direct simulations of the SDE.

that the "escape" probability is reduced significantly. As σ is varied in this modified problem, our methodology gives the following results, which we compare with probability estimates obtained by direct simulations of the stochastic differential equation in (40). When $\sigma = 1$, 98% of the simulations enter the region around the other equilibrium, whereas our methodology gives an upper bound of $\beta = 0.99$. When $\sigma = 0.5$, direct simulations gives 0.57, and our methodology returns $\beta = 0.6$. We see now that the expected result is obtained; as σ is decreased, the probability of reaching the other equilibrium (false switching) is decreased, and the upper bounds are close to the probability estimates obtained by direct simulations.

Using the same methodology, we can also address the dependence of the switching probability from one steady-state to the other on time and for a given noise intensity. We show the results of our computations in Figure 6 where the upper bound on this probability is plotted as a function of time, along with estimates of the probability, obtained from simulations of the SDE. It can be seen that the two plots are satisfactorily close to each other. However, while we had to run a large number of simulations (of order a thousand) to compute estimates of the probability, with the computational effort increasing as the simulation time interval was increased, the barrier technique generated good upper bounds in a fraction of time and each run had the same computational burden. This will become increasingly important as we consider larger systems where exact simulations become prohibitively slow. In general, probabilities computed using our barrier methods are conservative bounds on the true probabilities. However, as illustrated in our example, these bounds can yield accurate results.

The same techniques can again be used to compute probability lower bounds. For example, Figure 7 shows the upper and lower bounds for the probability of escaping a neighborhood of the high equilibrium in a finite time horizon as the noise intensity increases. It is interesting to note that the probability goes sharply from 0 to 1 (indicating a phase transition) as the noise is increased, and that the technique we propose can be used to estimate the critical noise intensity necessary for escaping the given region around the equilibrium.

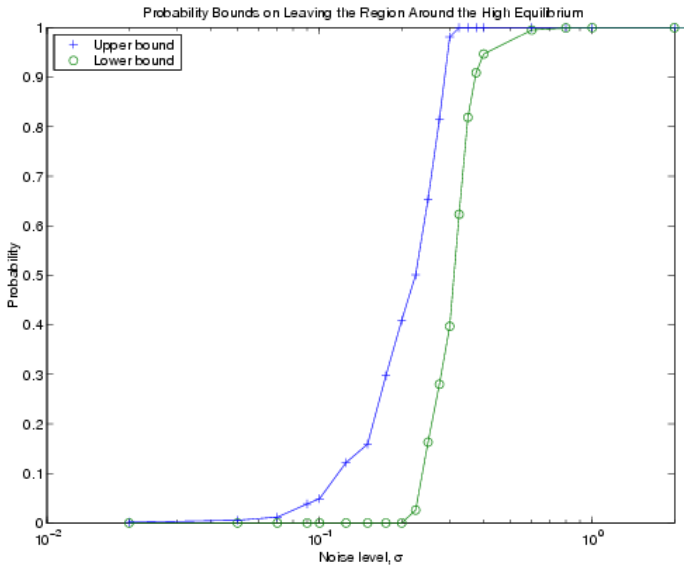

Figure 7: Upper and lower bounds on probability of escaping a neighborhood of the high equilibrium, computed using the Barrier technique.

The λ-Phage Two-Dimensional Model The model we have presented in the previous section is particularly simple. In order to demonstrate the capabilities of the methodology in higher dimensional systems, we propose to adopt a 2-D description of the λ system while giving a more plausible description of the biochemical noise affecting the system. In order to get a rich model description, we start by introducing a crucial step that was eliminated from the model to reduce it to a 1-D description: translation. Recall that in the previous model, the production of X follows

$$DX_2 + P \xrightarrow{k_t} DX_2 + P + nX, \quad X \xrightarrow{k_d} A$$

This means that X is produced in a single step process, then degraded. More realistically, it should be produced through a two step process of transcription first, then translation. The resulting description is

$$\begin{aligned} DX_2 + P &\xrightarrow{k_t} DX_2 + P + M, \quad M \xrightarrow{k_T} M + nX, \\ M &\xrightarrow{k_d} B, \quad X \xrightarrow{k_m} A \end{aligned} \tag{44}$$

and the resulting 2-D model is

$$\begin{aligned} \dot{m} &= k_t p_0 u - k_d m + q \\ \dot{x} &= n k_T m - k_m x + 2k_{-1} y - 2k_1 x^2 \end{aligned}$$

Proceeding as before, we eliminate u from the equation to get:

$$\begin{aligned} \dot{m} &= \frac{k_t p_0 d_t K_1 K_2 x^2}{1 + 2K_1 K_2 x^2 + 5K_1^2 K_2^2 x^4} - k_d m + q \\ \dot{x} &= n k_T m - k_m x \end{aligned}$$

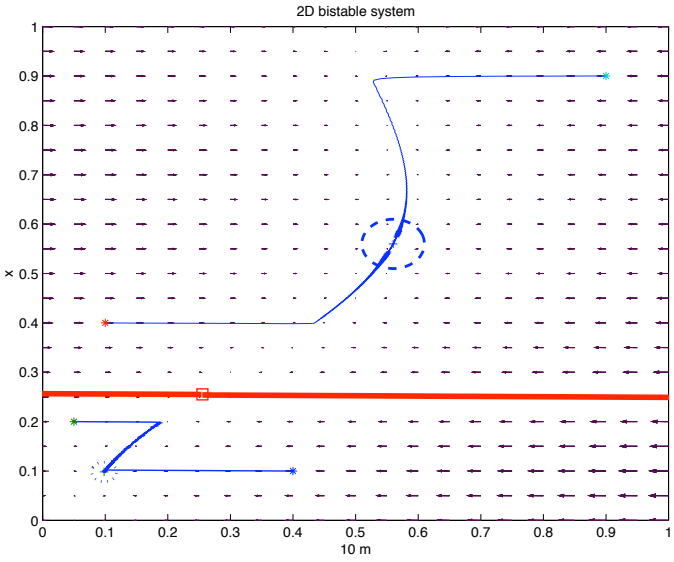

Figure 8: Phase plane for system (45–46). Arrows denote the vector field, solid lines are trajectories from initial conditions denoted by '*'. Equilibria are shown by '+' (stable) and '□' (unstable). The solid thick line is a separatix - it divides the phase plane in two, so that if the deterministic system is initialized in one region then all trajectories flow towards one equilibrium, whereas if the system is initialized in the other all trajectories flow to the other equilibrium.

Carrying the change of variables $\tilde{x} = x\sqrt{K_1K_2}$, $\tilde{m} = m\sqrt{K_1K_2}$, and $\tau = t(q\sqrt{K_1K_2})$, we get

$$\begin{aligned} \dot{\tilde{m}} &= \frac{\alpha\tilde{x}^2}{1+2\tilde{x}^2+5\tilde{x}^4} - \gamma\tilde{m} + 1 \\ \dot{\tilde{x}} &= \frac{nk_T}{q\sqrt{K_1K_2}}\tilde{m} - \frac{k_m}{q\sqrt{K_1K_2}}\tilde{x} \end{aligned}$$

where $\alpha = \frac{k_t p_0 d_t}{q}$ and $\gamma = \frac{k_d}{q\sqrt{K_1K_2}}$. Following [46], we fix the numerical values of the parameters in the system so as to get the following model

$$\dot{\tilde{m}} = \frac{50\tilde{x}^2}{1+2\tilde{x}^2+5\tilde{x}^4} - 150\tilde{m} + 1 \tag{45}$$

$$\dot{\tilde{x}} = 10\tilde{m} - \tilde{x} \tag{46}$$

Note that in the phase plane, the deterministic part of this system results in three equilibria as before, two of which are stable and the other unstable. The phase-plane of system (45–46) is shown in Figure 8.

Rather than introducing additive noise in an ad-hoc manner to this 2-D system, we consider more biologically plausible sources of noise that originate from intrinsic biochemical fluctuations. However, starting from a detailed account of the noise that stems from all the elementary chemical reactions in the network, we carry out two approximations that generate an SDE with multiplicative noise as the system's stochastic description.

For the system under study, the elementary (birth and death) chemical reactions are given by (36) and (44). Each of these elementary reaction steps is characterized by its probability of occurrence. This probability is in turn captured by the so called *propensity functions*, which are simple functions of the rate constants and concentration of the reactants. The description based on

the elementary reactions and their probability of occurrence generates a continuous-time, discrete-state Markov Chain that is often studied through Monte-Carlo simulations.

Chemical reactions, including the example that we are considering, often occur at drastically different time scales, therefore making analysis and simulation of the Markov chain description of the processes involved computationally challenging. To circumvent this stiffness, it is common to replace the elementary reaction with the so-called elementary-complex reactions. One elementary-complex reaction is formed by an aggregation of elementary reactions. Therefore, the probability of occurrence of an elementary-complex reaction is an involved function of the state of the system. One method to compute these probabilities is to use Michaelis-Menten and various other well established approximations for the deterministic description of the system, then use the resulting expressions to define the complex reactions and their propensities [99]. For the example considered here, this will translate into replacing the elementary reactions in (36) and (44), by complex birth and death reactions for $\tilde{m}$ and $\tilde{x}$ whose propensities are taken from the differential equations in (45) and (46).

With these elementary-complex propensities generating a reduced order stochastic description of the λ-phage system, we carry out a diffusion approximation that transforms the Markov-chain type of description of these molecular reactions into a stochastic differential equation. The premises of this procedure were recently re-investigated in the work of Gillespie [42], to which we refer the reader for details. Here, we only state the final result. If $a_i(X)$ are the propensity functions that determine transition probabilities, A is a column vector formed by these propensities, and S is the stoichiometry matrix, then the deterministic description of the system (as in (45) and (46)) is given by

$$\frac{dX}{dt} = SA(X)$$

and the SDE generated by the diffusion approximation is given by

$$dX = SA(X)dt + S\sqrt{\mathrm{diag}(A(X))}d\xi \tag{47}$$

With the elementary to elementary-complex and diffusion approximations in place, the final CLE describing the 2-D λ system becomes

$$\begin{aligned} d\tilde{m} &= \left(\frac{50\tilde{x}^2}{1+2\tilde{x}^2+5\tilde{x}^4} - 150\tilde{m} + 1\right)dt + \sqrt{\frac{50\tilde{x}^2}{q(1+2\tilde{x}^2+5\tilde{x}^4)}}d\xi_1 + \sqrt{\frac{150\tilde{m}}{q}}d\xi_2 + \frac{d\xi_3}{\sqrt{q}} \\ d\tilde{x} &= (10\tilde{m} - \tilde{x})\,dt + \sqrt{\frac{10\tilde{m}}{q}}d\xi_4 + \sqrt{\frac{\tilde{x}}{q}}d\xi_5 \end{aligned} \tag{48}$$

For this system, we will again compute a bound on the probability of reaching one of the steady-states from the other one during a fixed time. In particular, we consider whether from an initial region around the low equilibrium defined by

$$\mathcal{X}_0 = \{x \in \mathbb{R} | (x - 0.098)^2 + (10m - 0.098)^2 - 0.02^2 \leq 0\} \tag{49}$$

one can reach a region around the high equilibrium defined by

$$\mathcal{X}_u = \{x \in \mathbb{R} | (x - 0.560)^2 + (10m - 0.560)^2 - 0.05^2 \leq 0\}, \tag{50}$$

in a certain time span, say $t \in [0, 10]$ for different noise intensities. In this case, and for a fixed structure of the system, different noise intensities can be achieved by changing the number of

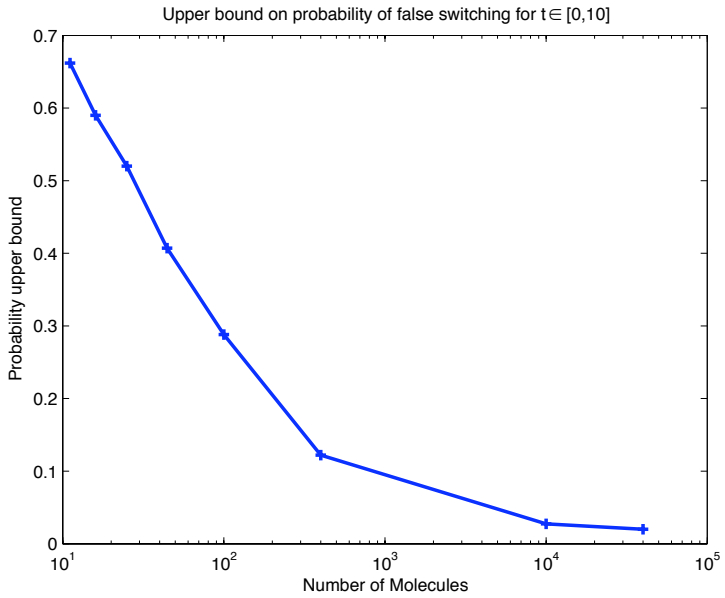

Figure 9: Upper bound on switching probability from the higher equilibrium as a function of the number of molecules in the system

molecules in the system. We therefore define $\hat{x} = Nx$ for equation (47). N is commonly called the system size. This change of variables scales the number of molecules without modifying the dynamics of the deterministic system. However, the associated noise intensity grows as the square root of N. This in agreement with the common intuition (and can also be directly seen from equation (47)) that as the number of molecules in a system increases, the noise strength affecting it, normalized by the mean, decreases.

With N as our varying noise intensity, we attempt to answer the false switching question. We construct certificates $B(x,t)$ algorithmically, again using the methodology described earlier. Note that the non-polynomial nature of the vector field (the way noise affects the system is through the square root of some polynomial) does not cause any problems, as these terms appear *squared* in the expressions to be tested — see condition (26) in Theorem 11. The upper bound on this probability as function of N is given in Figure 9. Notice the expected fact that as N increase, therefore yielding lower noise affecting the system, the probability of false switching decreases. As in the previous case, this result was obtained without time consuming computations and was not based on numerous simulations of the corresponding SDE.

3.1.3 Model Validation/Invalidation for the G-Protein Signaling in Yeast

Background and General Framework Let $u \in \mathbb{R}$, $y \in \mathbb{R}$ denote the input and output of a biological process, and $p \in \mathbb{R}^n$ be the vector of parameters. A model M of the process captures the relation between input and output as $y = M(u,p)$. Often, the model M arises from systems of ODEs of the form $\dot{x} = f(x,u,p),\ x(0) = x_0,\ y = g(x,u,p)$, where $x \in \mathbb{R}^{n_x}$ are the state variables (e.g., concentration of species). An experiment provides a dataset $(\nu,\ d,\ \delta,\ \epsilon)$, consists of input and output measurements ν and d, and their uncertainties δ and ϵ. Suppose prior knowledge on parameters can be expressed by L polynomials: $H = \{p \in \mathbb{R}^n : g_l(p) \leq 0, i = 1, \cdots, L\}$. The set of parameters consistent with m experiments is $P = \{p \in H :\ |M(u_i,p) - d_i| \leq \epsilon_i,\ |u_i - \nu_i| \leq \delta_i,\ i = 1, \ldots, m\}$. We want to either prove P is empty or describe P as precisely as we can. We start with searching

for an emptiness certificate using the barrier function methodology, as discussed later. If not empty, P can be described by solutions to a series of optimization problems of the form

$$\begin{array}{ll} \text{minimize} & \Phi(p) \\ \text{subject to} & p \in P, \end{array} \tag{51}$$

where we choose $\Phi(p)$ depending on the aspect of P we are interested in. In general we cannot express $M(u,p)$ in closed form and have no way of solving or relaxing these optimization problems. However, we can use SOS methods if we first approximate $M(u,p)$ by a polynomial $S(u,p)$ in the region of interest. S is referred to as the surrogate model [35,40]. The difference between M and S can be made arbitrarily small if the order of S is high enough. Different types of surrogate models can be considered; one is to use response surface methods to generate a quadratic surrogate model as in [40]. If we assume the approximation error is bounded on the feasible parameter set, i.e. $|M(u_i,p) - S(u_i,p)| \leq e_i,\ \forall p \in P,\ |u_i - \nu_i| \leq \delta_i,\ i = 1,\ldots,m$, then we can construct two sets P_I and P_O, which are sets of parameters consistent with output data within margins $\epsilon_i - e_i$ and $\epsilon_i + e_i$, respectively. Since $P_I \subset P \subset P_O$, $P_O = \emptyset$ is a sufficient condition for inconsistency between model and data, which can be checked using SOS methods. If $P \neq \emptyset$, we can use SOS to compute upper bounds on Φ^*, the optimal value of problem (51), as will be shown in the examples.

G-protein Model Consider the following model for the heterotrimeric G-protein cycle in yeast mediating the response to mating pheromone. Model includes the following processes: (1) the binding kinetics of ligand (L) to receptor (R); (2) the synthesis and degradation of receptor; (3) activation of G-protein (G) by active receptor (RL); (4) deactivation of G_α-GTP (G_α) catalyzed by the RGS protein Sst2p; and (5) reassociation of the heterotrimer. For several of the reactions, the nominal value for the rate constants or parameters have been measured directly: ($k_1 = 10^6 mM^{-1}s^{-1},\ k_2 = 10^{-2}s^{-1},\ k_3 = 4\times10^{-4}s^{-1},\ k_4 = 4\times10^{-3}s^{-1},\ k_5 = 4mMs^{-1},\ G_t = 10^4 mM$). For other reactions, these values were inferred from input-output data ($k_6 = 10^{-5}mM^{-1}s^{-1}$ and $k_7 = 0.1s^{-1}$), and for some they were based on estimates in the literature ($k_8 = 1mM^{-1}s^{-1}$). Thus we have the following ODE model in which the input is the pheromone ligand and the output is the (normalized) level of free $G_{\beta\gamma}$ (Gbg).

$$\begin{array}{rcl} \dot{x}_1 &=& -k_1x_1u + k_2x_2 - k_3x_1 + k_5 \\ \dot{x}_2 &=& k_1x_1u - k_2x_2 - k_4x_2 \\ \dot{x}_3 &=& -k_6x_2x_3 + k_8(G_t - x_3 - x_4)(G_t - x_3) \\ \dot{x}_4 &=& k_6x_2x_3 - k_7x_4 \\ y &=& (G_t - x_3)/G_t, \end{array} \tag{52}$$

where x_1=[R], x_2=[RL], x_3=[G], x_4=[G_α], u=[L], y=[$G_{\beta\gamma}$]/G_t, and G_t is the total amount of G-protein. The in-vivo dynamics and regulation of this cycle in yeast has been measured using fluorescence resonance energy transfer (FRET) [131]; we use the data from dose-response experiments: u = [1, 2, 5, 10, 20, 50, 100, 1000] (in nM), and y = [0.083, 0.122, 0.240, 0.352, 0.384, 0.397, 0.400, 0.397], where step input of size u is applied and output y is measured at $t = 60s$.

Invalidation Using Barrier Certificates We discuss the problem of invalidating the ODE model (52) with experimental data, using barrier certificates. Barrier certificates are discussed in section (3.1.1), where it is also noted that the search for a barrier can be cast as an SOS program. We applied this method to the ODE and data described above. One result is that we can invalidate the model using data from only the first experiment $u = 10^{-9},\ y = 0.038$, for relatively small uncertainty in the parameters and output. Specifically, we allowed for 4% uncertainty in k_6 and

k_7, the two parameters we are more uncertain about, and 5% uncertainty in the measured output, while fixing all other parameters and the input. Other a priori constraints on the states, typically coming from experiments, can also be handled. Here we used simulation to estimate bounds on the states, and included those constraints. We were then able to find a barrier certificate $B(x, t, p)$, a polynomial of degree 2 in x, 2 in t, and 1 in p, that invalidates the model with this experiment.

In this approach, there is no need to calculate surrogate models and deal with the extra error introduced by them, making this method more rigorous and less conservative. Also, uncertainty in the initial and final states and state constraints can be explicitly handled. The main disadvantage of the approach is the computational cost. Even though the size of the SOS program (for a fixed-degree barrier) is polynomial in the number of parameters, it grows very fast with the number of input/output data points. Currently, the only known way to include the data requires augmenting the state space, adding n_x states to the state space for each additional data point. This makes the method impractical for large amounts of data, and emphasizes the need for a different, more scalable method of handling data. Our research effort will attempt to address this issue.

Invalidation with Surrogate Models Quadratic surrogate models described before are used to prove that the outer set P_O is empty. This implies the emptiness of P, hence invalidating the original model M. The uncertainty in data and parameters are captured by I polynomials $\Psi_i(p, u)$. The set P_O is described as $P_O = \{p \in \mathbb{R}^n : \Psi_i(p, u) \leq 0, i = 1, \cdots, I\}$. Using SOS methods described earlier, if we can find nonnegative polynomials λ's of p and u such that $\sum_{i=1}^{I} \lambda_i \Psi_i(p, u) > 0$, for all values of p and u, then P_O is proven empty. These λ's constitute an invalidation certificate (in the special case that all constraints are quadratic and the multipliers are nonnegative constants, this condition is equivalent to the S-procedure). The values of these multipliers provide important information: constraints that have zero multipliers do not contribute to invalidating the model.

In the G-protein model, when the uncertainty is 10% on experiment data, 50% for k_6, k_7 and 4% for all other parameters, model (52) is invalidated by the experiment data given earlier with constant multipliers. Nonzero multipliers identify three data points (1, 7 and 8) that invalidate the model. In fact, model (52) is invalidated by only experiment data 1 and 8. The nonzero multipliers also point out directional information: the output measurement in the first experiment was too high and the eighth too low.

Descriptions of Feasible Parameter Space We have also used surrogate models to describe the multi-dimensional feasible parameter space. The simplest representation is to calculate upper and lower bounds on each parameter. Given that the measurement data invalidated the model, to study the feasible parameter space we used simulated data as described in [129]. We found that the simulated data do not add information to prior knowledge on ranges of individual parameters. This highlights the limitations of bounding parameter individually, and the importance of capturing parameter correlations. For example, the ratio between the G-protein activation rate k_6 and the deactivation rate k_7 is expected to be close to a constant since it determines the level of active G-protein. Using the simulated data and the prior bounds, we identified two parallel lines that bound the 2-dimensional feasible set from outside, as shown in Figure 10. The slopes and offsets of the lines were set as free variables in the optimization problem, and the vertical distance between the two lines was minimized. The feasible parameter set P is a subset of the shaded region.

In figure 11, the optimization is repeated as the slopes of the lines are varied over a range and vertical distance is minimized. This results in a set of hyperplanes bounding the feasible set. Such bounds can help describe the feasible parameter region, even in higher dimensions where they are not easy to visualize.

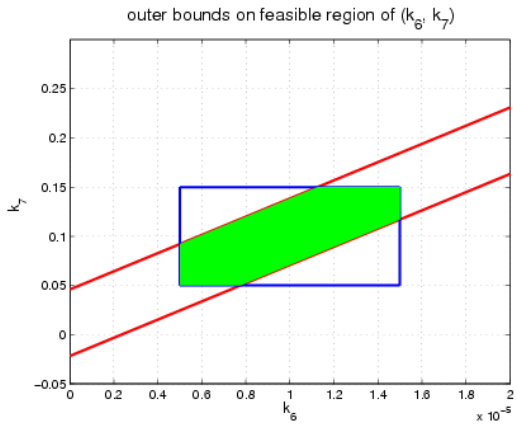

Figure 10: Shaded region includes the feasible parameter space allowed by data.

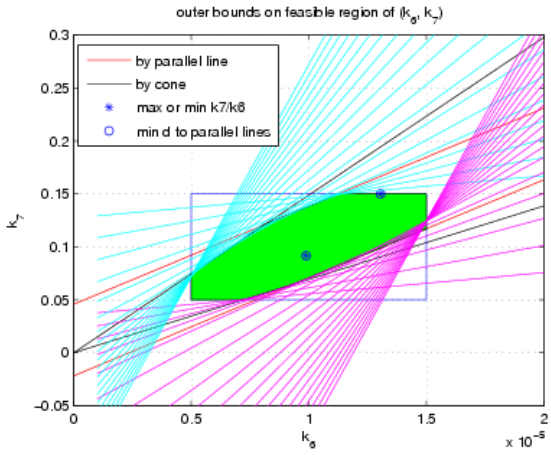

Figure 11: Lines with varying slopes bound the feasible parameter region.

3.1.4 Optimization-Based Methods for System Verification

A body of techniques based on convex optimization and sum of squares programming has been developed for verification of a large class of dynamical systems, including those with nonlinear dynamics, uncertainties, hybrid (mixed discrete-continuous) dynamics, stochasticity, and time-delay [85–87, 94, 95]. These techniques verify temporal properties such as safety (e.g., something bad never happens), reachability (e.g., something good can happen), eventuality/liveness (e.g., something good will surely happen), and their simple combinations, using certain functions of states called barrier certificates and density functions [85, 95]. The use of convex optimization gives several advantages on both the theoretical and computational sides. First, there is no need to propagate sets of states, and therefore the methods can be easily applied to a very large class of systems. This is even true for systems with infinite-dimensional state space (such as time-delay systems), whose verification had not been considered before in the literature. Not only that, convex duality can also be exploited to obtain converse statements. Finally, the computation of proofs can be performed using sum of squares programming, when the system has a polynomial description.

For a simple illustration, consider a continuous system $\dot{x} = f(x, d)$ where x is the state of the

system taking its value in the state space $\mathcal{X}$ and d is a disturbance input taking its value in $\mathcal{D}$. In addition, consider $\mathcal{X}_0 \subset \mathcal{X}$ as the set of possible initial states, and $\mathcal{X}_u \subset \mathcal{X}$ as the set of unsafe states. Suppose there exists a barrier certificate, i.e., a differentiable function $B : \mathcal{X} \to \mathbb{R}$ satisfying the inequalities

$$B(x) \leq 0 \quad \forall x \in \mathcal{X}_0, \tag{53}$$

$$B(x) > 0 \quad \forall x \in \mathcal{X}_u, \tag{54}$$

$$\frac{\partial B}{\partial x}(x) f(x, d) \leq 0 \quad \forall x \in \mathcal{X} \times \mathcal{D}. \tag{55}$$

Then it is easy to see that the safety property holds, i.e., that for all possible initial state $x_0 \in \mathcal{X}_0$ and for all possible disturbance input there exists no trajectory of the system that goes from the initial set to the unsafe set. Systems with hybrid dynamics can be treated in an analogous manner. Here we should ask that during the discrete transition the value of $B(x)$ also satisfies certain non-increasing conditions, similar to what we have in (55).

It is obvious that simulation is of limited use to address the verification of safety property stated above. Since the state of the system is uncountable, verifying by simulation that the property holds in all cases is never exact, simply because it is impossible to test all system behaviors. In fact, simulation alone may fail to uncover the existence of bad behaviors. Using barrier certificates and density functions to prove safety, reachability, and eventuality is analogous to using Lyapunov functions to prove stability. It eliminates the needs to run simulations, to explicitly compute the flow of the system, or to propagate sets of states.

For stochastic systems, such as those described by stochastic differential equations, safety verification can also be handled by computing an appropriate barrier certificate which upper-bounds the probability of reaching the unsafe set [87]. In this case, a barrier certificate $B : \mathcal{X} \to \mathbb{R}$ which generates a stochastic process $b(t) \triangleq B(x(t))$ that is a supermartingale, i.e., whose evolution along time is non-increasing on the *average*, is used. We also ask that the value of the barrier certificate at the initial states be lower than its value at the unsafe states. The probability of reaching the unsafe region can then bounded from above using a Chebyshev-like inequality for supermartingales.

There are other classes of systems that can be handled using this methodology. One example is given by time-delay systems. For verification of a time-delay system, a functional of states is used as a barrier certificate [86]. The forms of the functionals resemble the Lyapunov-Razumikhin functions or the Lyapunov-Krasovskii functionals used in stability analysis of time-delay systems. In [86], a hierarchy of functional structures is proposed to prove safety with decreasing levels of conservatism.

The conditions that must be satisfied by barrier certificates and density functions are formulated as convex programming problems. In addition to benefits in terms of computation, the duality structure inherent because of their formulation as convex programs also gives theoretical advantages [94, 95]. For example, the dual of safety verification, i.e., reachability verification, concerns proving the existence of a trajectory starting from the initial set that reaches another given set. Using insights from the linear programming duality appearing in the discrete shortest path problem, it is shown in [95] that reachability of continuous systems can also be verified through convex programming. Several convex programs for verifying safety and reachability, as well as other temporal properties such as eventuality, avoidance, and their combinations are formulated. As another example, a completeness statement in safety verification using barrier certificates is obtained by exploiting the strong duality between safety verification and reachability verification [94], stating that under reasonable technical conditions, the existence of a barrier certificate satisfying (53)–(55) is both sufficient and *necessary* for safety.

For systems and sets whose descriptions are in terms of polynomials, sum of squares programming described in Section 2.1 provides a hierarchy of scalable algorithmic methods for computing barrier certificates and density functions, where at each level the computational cost grows polynomially with respect to the system size. The computation can be performed efficiently using semidefinite programming, for example using the software SOSTOOLS. Because of the possibility to use sum of squares programming for computing barrier certificates and density functions, the methodology seems to be more scalable than many other existing methods that can handle nonlinear continuous and hybrid systems. Successful application of the method for verifying the safety property of a NASA life support system, which is a nonlinear hybrid systems with 6 discrete modes and 10 continuous states, has been reported in [43].

3.1.5 Stochastic Simulation

Our goal has been the development of a multiscale computational framework for the numerical simulation of chemically reacting systems, where each reaction will be treated at the appropriate scale, accounting for both stochastics and stiffness. Substantial progress was made in FY04. In microscopic systems formed by living cells, small numbers of reactant molecules can result in dynamical behavior that is discrete and stochastic rather than continuous and deterministic. In simulating and analyzing such behavior it is essential to employ methods that directly take into account the underlying discrete stochastic nature of the molecular events. This leads to an accurate description of the system that in many important cases is impossible to obtain through deterministic continuous modeling (e.g. ODEs). Gillespie's Stochastic Simulation Algorithm (SSA) has been widely used to treat these problems. However as a procedure that simulates every reaction event, it is prohibitively inefficient for most realistic problems. Our goal has been the development of a multiscale computational framework for the numerical simulation of chemically reacting systems, where each reaction will be treated at the appropriate scale. The framework is based on a sequence of approximations ranging from SSA at the smallest scale, through a "birth-death" Markov process approximation that is currently in progress, Gillespie's recently-developed tau-leaping approximation, a continuous stochastic differential equation (SDE) approximation, and finally to the familiar reaction rate equations (ODEs) at the coarsest scales. The Petzold-Gillespie collaboration has been very productive in FY04 in pursuing these goals. Listed below are papers issuing from this collaboration which were either published, accepted for publication, or submitted for publication during this period.

1. "Improved leap-size selection for accelerated stochastic simulation" by D. Gillespie and L. Petzold, published in *J. Chem. Phys.* 116:8229-8234 (Oct 2003). This paper developed a method for predicting the changes in the propensity functions during a finite time step that is more accurate than the method used in Gillespie's original tau-leaping paper [*J. Chem. Phys.* 115:1716, 2001]. This new method enables us to choose the size of the time step in a way that results in a reduced simulation time for a given level of accuracy.

2. "Stiffness in stochastic chemically reacting systems: the implicit tau-leaping method" by M. Rathinam, L. Petzold, Y. Cao, and D. Gillespie, published in *J. Chem. Phys.* 119:12784-12794 (Dec 2003). This paper examined the way in which "dynamical stiffness", which is well known in the context of deterministic differential equations, manifests itself in a stochastic context. Stiffness is the presence in the system of a wide range of dynamical modes, the fastest of which is stable. The paper shows that the original "explicit" tau-leaping procedure of Gillespie is severely limited in step-size when stiffness is present (a very common circumstance). The paper then proposes an "implicit" version of tau-leaping, and shows that for stiff systems

implicit tau-leaping can take significantly longer steps for a given level of accuracy than explicit tau-leaping can. The paper also shows that stiffness manifests itself more prominently in stochastic systems than in deterministic systems: the natural fluctuations keep moving the system off of the "slow manifold" and the stiffness keeps pulling the system back onto that manifold, resulting in an enhanced fluctuation signature.

3. "Efficient formulation of the stochastic simulation algorithm for chemically reacting systems" by Y. Cao, H. Li, and L. Petzold, accepted for publication by *J. Chem. Phys.* This paper develops a new way of implementing Gillespie's original "direct method" of doing stochastic simulations, which is as fast or faster than the "next reaction method" of Gibson and Bruck [J. Phys. Chem A 104:1876, 2000]. The essence of the method is to dynamically re-index the reaction channels so that those with larger propensity functions have smaller indices. This reduces the average time it takes for the algorithm to "look up" the identity of the next reaction that occurs. ¡/p¿

4. "Stochastic chemical kinetics" by D. Gillespie. Accepted for publication later this year as a chapter for a forthcoming *Handbook for Materials Modeling* edited by H. Metiu of UCSB and S. Yip of MIT. This offering is a concise but detailed tutorial on stochastic chemical kinetics. Gillespie this year gave an abbreviated version of this tutorial to the DARPA BioCOMP PI meeting in Adelphi MD, and later gave longer versions to three special seminars at MIT, UMBC, and MathWorks. MathWorks is interested in incorporating our simulation schemes into their popular commercial software product MatLab, so this represents technology transfer.

5. "Consistency and stability of tau-leaping schemes for chemical reaction systems" by M. Rathanam, L. Petzold, Y. Cao, and D. Gillespie. Submitted to *SIAM Multiscale Modeling and Simulation.* This paper develops a theory of local errors (single-step errors) for explicit and implicit tau-leaping. The results establish that both tau-leaping methods are first-order consistent. The local error formulas derived in this paper could serve as a basis for future stepsize control techniques. The paper also shows that, in the special case of systems with linear propensity functions, both tau-leaping methods are first-order convergent in all moments. Finally, the paper shows that the implicit tau-leaping method (unlike the explicit tau-leaping method) is unconditionally stable in the mean for stable systems.

6. "The slow-scale stochastic simulation algorithm" by Y. Cao, D. Gillespie, and L. Petzold, submitted to *J. Chem. Phys.* This paper introduces what we believe is a very promising approach to the "multiscale problem" in chemical kinetics. That problem is the fact that reactions in real chemical systems often take place on vastly different time scales, with "fast" reactions firing much more frequently than "slow" ones. Although the fast reactions are usually much less important than the slow ones, the exact stochastic simulation algorithm (SSA) necessarily spends most of its time simulating the fast reactions. Is there a way to skip over the unimportant fast reactions, and directly simulate only the important slow ones? This paper gives an affirmative answer to that question, an answer that we believe is more complete and easier to understand and use than related approaches proposed by E. Haseltine and J. Rawlings [J. Chem. Phys. 117:6959, 2002] and C. Rao and A. Arkin [J. Chem. Phys. 118:4999), 2003]. The key is a new theorem, formulated and proved in this paper, which shows how to construct a modified form of the propensity function of each slow reaction. The ability to do this leads directly to a way of stochastically stepping from one slow reaction to the next, an algorithm that we have named the "slow-scale SSA". The ssSSA is shown to work well

for two simple model stiff systems, giving real-time simulation speedups of over 1000 with no significant loss of simulation accuracy. In the course of this work, a theoretical explanation was found (in the proof of our theorem) for a curious "low bandpass filtering effect" that was earlier noted by Doyle, Khammash, and coworkers in the heat-shock response mechanism of the E. coli bacterium. Future work on the ssSSA will be aimed at developing ways of applying it to more complicated systems. The ssSSA is far superior to explicit tau-leaping for stiff systems, and we believe that future work will reveal an enlightening connection to implicit tau-leaping.

7. "Multiscale stochastic simulation algorithm with stochastic partial equilibrium assumption for chemically reacting systems" by Y. Cao, D. Gillespie, and L. Petzold. Submitted to *J. Computational Phys.* This paper describes an extended version of the theory outlined in #6 that emphasizes the close connection to the traditional "partial equilibrium" approximation in deterministic chemical kinetics. The technique of "down-shifting", first reported in the paper of #2, is shown to be useful for "relaxing" the fast variables to their asymptotic stationary distribution. The MSSA technique is applied to several model systems, including a modeling of the heat shock response mechanism in E. coli that involves 28 species and 61 reactions. Simulation time speed ups over the exact SSA of nearly 2 orders of magnitude are achieved without significant loss of accuracy.

8. "Trapezoidal tau-leaping formula for the stochastic simulation of chemically reacting systems" by Y. Cao and L. Petzold. Being written. This paper introduces a new leaping formula, which reduces to the trapezoidal rule in the deterministic regime. Numerical experiments demonstrate that this new formula has better accuracy and stability than both the explicit and implicit tau formulas.

9. "The numerical stability of leaping methods for stochastic simulation of chemically reacting systems" by Y. Cao, L. Petzold, M. Rathinam, and D. Gillespie. Being written. This paper discusses the consequences of the stiffness for stochastic simulation using leaping methods. The numerical stability of three tau-leaping formulas (explicit, implicit, and trapezoidal) are discussed and compared.

10. "Accuracy limitations and the measurement of errors in the stochastic simulation of chemically reacting systems", Y. Cao and L. Petzold. Being written. In stochastic simulation using non-exact methods, a common problem is to determine the accuracy of a proposed stepping formula. This paper discusses how to measure the error in distribution. "Kolmogorov distance" and "histogram distance" are presented and compared. Some accuracy bounds are given for these measurements. These bounds quantitatively give the "round-off" error in stochastic simulations, which we believe will prove to be a widely useful concept. Finally it should be noted that Gillespie is serving as an auxiliary advisor to a student of Petzold, Sotiria Lampoudi, who has just begun to look at some problems associated with relaxing the "well-stirred" assumption that underlies the conventional chemical master equation and SSA. The specific question being examined by Lampoudi is which of two seemingly well founded but conflicting expressions for the diffusive transport coefficient (between two contiguous spatially homogeneous system subvolumes) is really correct. This is a first step in addressing the complicated problem of simulating systems which are not spatially homogeneous.

3.1.6 Sum of Squares (SOS) Framework and SOSTOOLS

Substantial progress was made in FY04 on model (in)validation from data, hybrid system verification with applications to analysis of gene regulatory networks, overcoming intractability of coNP hard problems by exploiting robustness, and in applications to specific biological problems, but particularly beat shock in E. Coli. The latter is in collaboration with Mustafa Khammash (UCSB) and Carol Gross (UCSF) and the experimental and modeling work is funded by NSF, but uses SOSTOOLS capabilities partially developed with DARPA funding.

The aim is build on mathematics of systems engineering to create a coherent and complete theoretical infrastructure proceeding from experimental data to modeling, analysis, inference, and with tight feedback to experimentation and modeling throughout. Both data and modeling assertions and questions must be described in a common framework that is biologically natural, yet can be turned over to powerful algorithms for resolution. Is a model consistent with experimental data, which may come from extremely heterogeneous sources? If so, is it robust to additional perturbations that are plausible but untested? Are different models at multiple scales of resolution consistent? What is the most promising experiment to refute a model? Put in natural terms (which are typically nonlinear, nonequilibrium, uncertain, stochastic, hybrid and so on), such questions are conventionally viewed as computationally intractable, and biologists have traditionally been forced to translate into unnatural terms in order to use available algorithms. This is undesirable and now potentially unnecessary.

Our new mathematics involves blending and augmenting tools from dynamical systems, control theory, and operator theory in such a way that natural biological questions can be recast as statements about real semi-algebraic sets. Proving such statements is generally coNP-hard, but real algebraic geometry, semi-definite programming, and duality theory from optimization provide new methods to systematically exhaust coNP by searching for nested families of short proofs using convex relaxations. This has been used to create systematic nested families of relaxations which directly answer all of the above problems for classes of problems that appear to include biological networks. Applications to a variety of "toy" biologically motivated problems have been used to test the algorithms and several experimentally meaningful biological problems have successfully been tackled as well. A crucial insight is that evolution favors high robustness to uncertain environments and components, yet allows severe fragility to novel perturbations, and this "robust yet fragile"' feature must be exploited explicitly in scalable algorithmic approaches. Substantial progress has been made in doing this.

Specifically, the new modeling and inference framework we are developing need not handle arbitrary problems of the type above (presumably worst-case intractable) but only that subset of problems which are biologically meaningful. Biological organisms are highly constrained in that they have not just evolved, but necessarily evolved in ways that are robust to uncertainties in their environment and their component parts. These are extremely severe constraints, not present in other sciences but essential in engineering, which emerge primarily at the network level in both biology and engineering. These robustness constraints play a crucial if often hidden and implicit role in the informal processes that biologists use to reason about their experiments and are central in creating a scalable process of biological inference. A theoretical and software infrastructure that does not explicitly exploit the highly structured, evolved, and "robust yet fragile" nature of biological systems is hopelessly doomed to be overwhelmed by their sheer complexity. A key insight is that high computational complexity implies the existence of hidden fragilities in the original problem statements, models, or the data, that must be resolved either by refined modeling or new experiments. Put simply, "complexity implies fragility" and robust systems can be tractably modeled and understood, with the proper infrastructure. In our framework, lack of short proofs

or simple relaxations implies, by a generalization of duality, that there are inherent fragilities of the type described above. In other words, "dual complexity implies primal fragility." This feedback does not imply that P=NP=coNP, which is unlikely, but that the inference problems within coNP lacking short proofs can be traced to biologically meaningful flaws in models or data for which resolution can then be systematically pursued. While all the implications of these results have yet to be explored, it is already clear that these insights are essential for creating a scalable computational infrastructure for systems biology.

There have been a large number of publications related to this work published or submitted in FY04. The first is an application to a biological problem, and the rest are mathematical foundations of relevance to biology

1. H. El-Samad, S. Prajna, A. Papachristodoulou, M. H. Khammash, and J. C. Doyle. "Model Validation and Robust Stability Analysis of the Bacterial Heat Shock Response Using SOSTOOLS." In *Proceedings of the IEEE Conference on Decision and Control*. Maui, HI. 2003. The complexity inherent in gene regulatory network models, as well as their nonlinear nature make them difficult to analyze or validate/invalidate using conventional tools. Combining ideas from robust control theory, real algebraic geometry, optimization and semi-definite programming, SOSTOOLS provides a promising framework to answer these robustness and model validation questions algorithmically. We adopt these tools in the study of the heat shock response in bacteria. For this purpose, we use a reduced order model of the bacterial heat stress response. We study the robust stability properties of this system to parametric uncertainty, and address the model validation/invalidation problem by proving the necessity for the existence of certain feedback loops to reproduce the known time behavior of the system.

2. S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo. "SOSTOOLS and its Control Applications." To appear in *Positive Polynomials in Control*. A. Garulli and D. Henrion (eds.). Springer-Verlag. 2004. In this chapter we present SOSTOOLS, a third-party MATLAB toolbox for formulating and solving sum of squares optimization problems. Sum of squares optimization forms a basis for formulating convex relaxations to computationally hard problems such as some that appear in systems and control. Currently, sum of squares programs are solved by casting them as semidefinite programs, which can in turn be solved using interior-point based numerical methods. SOSTOOLS helps this translation in such a way that the underlying computations are abstracted from the user. Here we give a brief description of the toolbox, its features and capabilities (with emphasis on the recently added ones), as well as show how it can be applied to solving problems of interest in systems and control.

3. A. Papachristodoulou and S. Prajna. "Analysis of Non-polynomial Systems using the Sum of Squares Decomposition." To appear in *Positive Polynomials in Control*. A. Garulli and D. Henrion (eds.). Springer-Verlag. 2004. Recent advances in semidefinite programming along with use of the sum of squares decomposition to check nonnegativity have paved the way for efficient and algorithmic analysis of systems with polynomial vector fields. In this paper we present a systematic methodology for analyzing the more general class of non-polynomial vector fields, by recasting them into rational vector fields. The sum of squares decomposition techniques can then be applied in conjunction with an extension of the Lyapunov stability theorem to investigate the stability and other properties of the recasted systems, from which properties of the original, non-polynomial systems can be inferred. This will be illustrated by some examples from the mechanical and chemical engineering domains.

4. S. Prajna and A. Jadbabaie. "Safety Verification of Hybrid Systems using Barrier Certificates." In *Hybrid Systems: Computation and Control*. R. Alur and G. J. Pappas (eds). Springer–Verlag. 2004. This paper presents a novel methodology for safety verification of hybrid systems. For proving that all trajectories of a hybrid system do not enter an unsafe region, the proposed method uses a function of state termed a barrier certificate. The zero level set of a barrier certificate separates the unsafe region from all possible trajectories starting from a given set of initial conditions, hence providing an exact proof of system safety. No explicit computation of reachable sets is required in the construction of barrier certificates, which makes nonlinearity, uncertainty, and constraints can be handled directly within this framework. The method is also computationally tractable, since barrier certificates can be constructed using the sum of squares decomposition and semidefinite programming. Some examples are provided to illustrate the use of the method.

5. S. Prajna, A. Rantzer. "On Homogeneous Density Functions." In *Directions in Mathematical Systems Theory and Optimization*. A. Rantzer and C. I. Byrnes (eds). Springer–Verlag. 2003. We consider homogeneous density functions for proving almost global attractivity of the zero equilibrium in a homogeneous system. It is shown that the existence of such a function is guaranteed when the equilibrium is asymptotically stable, or in the more general case, when there exists a nonhomogeneous density function for the same system satisfying some reasonable conditions. Results related to robustness under nonhomogenizing perturbations are also presented.

6. A. Papachristodoulou and S. Prajna. "Nonlinear Systems Analysis using the Sum of Squares Decomposition." Submitted to *IEEE Transactions on Automatic Control*. 2004. The algorithmic analysis of nonlinear systems has always been a challenging task. A methodology based on the sum of squares decomposition of multivariate polynomials and semidefinite programming has been proposed recently to efficiently construct Lyapunov functions for proving stability of nonlinear systems with polynomial vector fields. In this paper, we extend the methodology to handle systems that evolve under equality, inequality and integral constraints. This allows analysis of systems described by differential algebraic equations (DAEs), robustness analysis under parametric or dynamic uncertainty, and analysis of systems with non-polynomial vector fields. We also show how input-output and other types of analysis can be performed using the same methodology. The use of the proposed technique is illustrated by various examples.

7. S. Prajna. "Barrier Certificates for Nonlinear Model Validation." Submitted to *Automatica*. 2003. New methods for model validation of continuous-time nonlinear systems with uncertain parameters are presented in this paper. The methods employ functions of state-parameter-time, termed here as barrier certificates, whose existence proves that a model and a feasible parameter set are inconsistent with some time-domain experimental data. A very large class of models can be treated within this framework; this includes differential-algebraic models, models with memoryless/dynamic uncertainties, and hybrid models. Construction of barrier certificates can be performed by convex optimization, utilizing recent results on the sum of squares decomposition of multivariate polynomials.

8. F. Wu and S. Prajna. "SOS-based Solution Approach to Polynomial LPV System Analysis and Synthesis Problems." Submitted to *International Journal of Control*. 2003. Based on Sum-of-Squares (SOS) decomposition, we propose a new solution approach for polynomial LPV system analysis and control synthesis problems. Instead of solving matrix variables

over positive cone, the SOS approach tries to find a suitable decomposition to verify the positiveness of given polynomials. The complexity of the SOS-based numerical method is polynomial of the problem size, and is computationally attractive. This approach also leads to more accurate solutions to LPV systems than most existing relaxation methods. Several examples have been used to demonstrate benefits of the SOS-based solution approach.

9. S. Prajna, A. Jadbabaie, and G. J. Pappas. "Stochastic Safety Verification using Barrier Certificates." To appear in *Proceedings of the IEEE Conference on Decision and Control*. 2004. We develop a new method for safety verification of stochastic systems based on functions of states termed barrier certificates. Given a stochastic continuous or hybrid system, a set of initial states, and a set of unsafe states, our method computes an upper bound on the probability that a trajectory of the system reaches the unsafe set, a bound whose validity is proven by the existence of a barrier certificate. Both the upper bound and its corresponding barrier certificate can be computed using convex optimization, and hence the method is computationally tractable.

10. H. Yazarel, S. Prajna, and G. J. Pappas. "SOS for Safety." To appear in *Proceedings of the IEEE Conference on Decision and Control*. 2004. One of the difficult problems for hybrid systems is the safety verification problem which asks whether trajectories starting from a set of initial states reach a set of an unsafe (final) set. Verification of continuous systems remains one of the main obstacles in the safety verification of hybrid systems. In this paper, by exploiting the structure of linear dynamical systems, we pose the exact safety verification of linear systems with certain eigen-structure as emptiness of a set defined by polynomial equalities and inequalities. Sum of squares (SOS) decomposition is then employed to check emptiness of the set, which can effectively be computed by semidefinite programming.

11. S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo. "SOSTOOLS: Control Applications and New Developments." To appear in *Proceedings of the IEEE Conference on Computer Aided Control Systems Design*. 2004. In this paper we describe recent developments and control applications of SOSTOOLS, a free third-party MATLAB toolbox for formulating and solving sum of squares programs.

12. S. Prajna, A. Papachristodoulou, and F. Wu. "Nonlinear Control Synthesis by Sum of Squares Optimization: A Lyapunov-based Approach." In *Proceedings of the Asian Control Conference*. 2004. This paper addresses the state feedback control synthesis problems for nonlinear systems, either without or with guaranteed cost or H_∞ performance objectives. By representing the open-loop nonlinear systems in a state dependent linear-like form and considering a special class of Lyapunov functions, sufficient conditions for the solutions to the above problems can be formulated in terms of state dependent linear matrix inequalities. Semidefinite programming relaxations based on the sum of squares decomposition are then used to efficiently solve such inequalities.

13. S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo. "New Developments in Sum of Squares Optimization and SOSTOOLS." In *Proceedings of the American Control Conference*. 2004. We describe the latest additions to SOSTOOLS, a freely available MATLAB toolbox for formulating and solving sum of squares programs. Among the many improvements, there are native polynomial objects, structure-exploiting techniques for sparse and structured polynomials, new customized functions, and support for alternative SDP solvers. We sketch some of the theory behind the new improvements, and illustrate the new commands using control-oriented examples.

14. F. Wu and S. Prajna. "A New Solution Approach to Polynomial LPV System Analysis and Synthesis." In *Proceedings of the American Control Conference*. 2004. Based on Sum-of-Squares (SOS) decomposition, we propose a new solution approach for polynomial LPV system analysis and control synthesis problems. Instead of solving matrix variables over a positive definite cone, the SOS approach tries to find a suitable decomposition to verify the positiveness of given polynomials. The complexity of the SOS-based numerical method is polynomial of the problem size. This approach also leads to more accurate solutions to LPV systems than most existing relaxation methods. Several examples have been used to demonstrate benefits of the SOS-based solution approach.

15. S. Prajna. "Barrier Certificates for Nonlinear Model Validation." In *Proceedings of the IEEE Conference on Decision and Control*. Maui, HI. 2003. New methods for model validation of continuous-time nonlinear systems with uncertain parameters are presented in this paper. The methods employ functions of state-parameter-time, termed here as barrier certificates, whose existence proves that a model and a feasible parameter set are inconsistent with some time-domain experimental data. A very large class of models, including differential-algebraic models, models with memoryless/dynamic uncertainties, and hybrid models, can be treated within this framework. Construction of barrier certificates can be performed by convex optimization, utilizing the sum of squares decomposition of multivariate polynomials.

16. S. Prajna. "POD Model Reduction with Stability Guarantee." In *Proceedings of the IEEE Conference on Decision and Control*. Maui, HI. 2003. Issues concerning stability of models obtained from model reduction using the proper orthogonal decomposition (POD) are investigated in this paper. A sufficient condition which guarantees preservation of stability is given, and a stability-preserving POD model reduction scheme is proposed.

17. S. Prajna and A. Papachristodoulou. "Analysis of Switched and Hybrid Systems – Beyond Piecewise Quadratic Methods." In *Proceedings of the American Control Conference*. Denver, CO. 2003. This paper presents a method for stability analysis of switched and hybrid systems using polynomial and piecewise polynomial Lyapunov functions. Computation of such functions can be performed using convex optimization, based on the sum of squares decomposition of multivariate polynomials. The analysis yields several improvements over previous methods and opens up new possibilities, including the possibility of treating nonlinear vector fields and/or switching surfaces and parametric robustness analysis in a unified way.

18. S. Prajna, P. A. Parrilo, and A. Rantzer. "Nonlinear Control Synthesis by Convex Optimization." *IEEE Transactions on Automatic Control* 49(2):310–314. 2004. A stability criterion for nonlinear systems, recently derived by the third author, can be viewed as a dual to Lyapunov's second theorem. The criterion is stated in terms of a function which can be interpreted as the stationary density of a substance that is generated all over the state space and flows along the system trajectories towards the equilibrium. The new criterion has a remarkable convexity property, which in this paper is used for controller synthesis via convex optimization. Recent numerical methods for verification of positivity of multivariate polynomials based on sum of squares decompositions are used.

3.2 SBML

3.2.1 Continued Development of *libSBML*

Expand the Model Consistency Checking Rules and Facilities Although libSBML consistency checks are not expressed directly in OCL, we have created an OCL-like language on top of the libSBML C++ API. This language balances the readability of OCL with the efficiency and expressiveness of C++, which is sometimes necessary for more complicated validation procedures. The language allows the manipulation of only `const`ant C++ objects, which much like OCL, guarantees operations will be side-effect free. Further, this guarantee is enforced at compile time. Being side-effect free is an important property as we do not want the process of consistency checking to change the state of a model. An example will help make these concepts more clear.

One of the 50 consistency checks currently implemented ensures that if a model author overrides the default definition of the *substance* unit, a special unit name in SBML, the resulting unit definition is consistent with the notion of a substance. The consistency check constraint is written as:

```
START_CONSTRAINT (1202, UnitDefinition, ud)
{
  msg =
   "A 'substance' UnitDefinition must simplify to a single "
   "Unit of kind 'mole' or 'item' with an exponent of '1' "
   "(L2v1 Section 4.4.3).";

  pre( ud.getId() == "substance" );

  inv( ud.getNumUnits() == 1                              );
  inv( ud.getUnit(0).isMole() || ud.getUnit(0).isItem() );
  inv( ud.getUnit(0).getExponent() == 1                   );
}
END_CONSTRAINT
```

The `START_CONSTRAINT` macro takes three arguments. The first is a number that uniquely identifies this constraint (i.e., 1202). Assigning such identifiers to each constraint facilitates traceability and allows programmers to easily determine which rules have been violated. The next two parameters indicate the type of SBML object to which this rule applies (i.e., **UnitDefinition**) and a shorthand name to use for the object being checked (i.e., `ud`).

The body of the constraint consists of a message (`msg`) to be logged should the SBML object fail the check. After the message, zero or more preconditions (`pre`) may be listed. In order for the rule to apply to the SBML object in question, all preconditions must hold (in the order listed). If a precondition does not hold, the check is aborted without logging either a passage or failure. Finally, assuming all preconditions hold, the object's state must adhere to a set of one or more invariants (`inv`). Should any invariant fail, the constraint immediately fails and a message is logged.

In the above example, notice that preconditions and invariants are specified on the (lib)SBML object model. Each method invocation (operation) does not change the state of the model and specifies *what* not *how* (with apologies made for the standard names used for getter methods, e.g. `getUnit()`, which arguably describes *how* and not *what*; even OCL falls victim to this slight, purely esthetic inconsistency.)

In the course of this project, we implemented all of the model consistency-checking rules originally envisioned, plus additional checks added during the development of SBML Level 2 Version 2, which appeared during the time of this project. The total number of consistency-checking rules is 126. The SBML Level 2 Version 2 specification document lists these checks in Appendix C, using

a new numbering scheme introduced in Version 2. For the reader's convenience, these rules are reproduced below:

General XML validation

10101. An SBML XML file must use UTF-8 as the character encoding. More precisely, the `encoding` attribute of the XML declaration at the beginning of the XML data stream cannot have a value other than "`UTF-8`". An example valid declaration is `<?xml version="1.0" encoding="UTF-8"?>`.

10102. An SBML XML document must not contain undefined elements or attributes in the SBML namespace. Documents containing unknown elements or attributes placed in the SBML namespace do not conform to the SBML Level 2 specification.

General MathML validation

10201. All MathML content in SBML must appear within a `math` element, and the `math` element must be either explicitly or implicitly in the XML namespace "`http://www.w3.org/1998/Math/MathML`".

10202. The only permitted MathML 2.0 elements in SBML Level 2 are the following: `cn`, `ci`, `csymbol`, `sep`, `apply`, `piecewise`, `piece`, `otherwise`, `eq`, `neq`, `gt`, `lt`, `geq`, `leq`, `plus`, `minus`, `times`, `divide`, `power`, `root`, `abs`, `exp`, `ln`, `log`, `floor`, `ceiling`, `factorial`, `and`, `or`, `xor`, `not`, `degree`, `bvar`, `logbase`, `sin`, `cos`, `tan`, `sec`, `csc`, `cot`, `sinh`, `cosh`, `tanh`, `sech`, `csch`, `coth`, `arcsin`, `arccos`, `arctan`, `arcsec`, `arccsc`, `arccot`, `arcsinh`, `arccosh`, `arctanh`, `arcsech`, `arccsch`, `arccoth`, `true`, `false`, `notanumber`, `pi`, `infinity`, `exponentiale`, `semantics`, `annotation`, and `annotation-xml`.

10203. In the SBML subset of MathML 2.0, the MathML attribute `encoding` is only permitted on `csymbol`. No other MathML elements may have the `encoding` attribute.

10204. In the SBML subset of MathML 2.0, the MathML attribute `definitionURL` is only permitted on `csymbol`. No other MathML elements may have a `definitionURL` attribute.

10205. In SBML Level 2, the only values permitted for the `definitionURL` attribute on a `csymbol` element are "`http://www.sbml.org/sbml/symbols/time`" and "`http://www.sbml.org/sbml/symbols/delay`".

10206. In the SBML subset of MathML 2.0, the MathML attribute `type` is only permitted on the `cn` construct. No other MathML elements may have a `type` attribute.

10207. The only permitted values for the `type` attribute on MathML `cn` elements are "`e-notation`", "`real`", "`integer`", and "`rational`".

10208. MathML `lambda` elements are only permitted as the first element inside the `math` element of a **FunctionDefinition**; they may not be used elsewhere in an SBML model.

10209. The arguments of the MathML logical operators `and`, `or`, `xor`, and `not` must have boolean values.

10210. The arguments to the following MathML constructs must have a numeric type: `plus`, `minus`, `times`, `divide`, `power`, `root`, `abs`, `exp`, `ln`, `log`, `floor`, `ceiling`, `factorial`, `sin`, `cos`, `tan`, `sec`, `csc`, `cot`, `sinh`, `cosh`, `tanh`, `sech`, `csch`, `coth`, `arcsin`, `arccos`, `arctan`, `arcsec`, `arccsc`, `arccot`, `arcsinh`, `arccosh`, `arctanh`, `arcsech`, `arccsch`, `arccoth`.

10211. The values of all arguments to `eq` and `neq` operators should have the same type (either all boolean or all numeric).

10212. The types of values within `piecewise` operators should all be consistent: the set of expressions that make up the first arguments of the `piece` and `otherwise` operators within the same `piecewise` operator should all return values of the same type.

10213. The second argument of a MathML `piece` operator must have a boolean value.

10214. Outside of a **FunctionDefinition**, if a `ci` element is the first element within a MathML `apply`, then the `ci`'s value can only be chosen from the set of identifiers of **FunctionDefinition**s defined in the SBML model.

10215. Outside of a **FunctionDefinition**, if a `ci` element is not the first element within a MathML `apply`, then the `ci`'s value can only be chosen from the set of identifiers of **Species**, **Compartment**, **Parameter** or **Reaction** objects defined in the SBML model.

10216. The `id` value of a **Parameter** defined within a **KineticLaw** can only be used in `ci` elements within the MathML content of that same **KineticLaw**; the identifier is not visible to other parts of the model.

10217. The MathML formulas in the following elements must yield numeric expressions: `math` in **KineticLaw**, `stoichiometryMath` in **SpeciesReference**, `math` in **InitialAssignment**, `math` in **AssignmentRule**, `math` in **RateRule**, `math` in **AlgebraicRule**, and `delay` in **Event**, and `math` in .

General identifier validation

10301. The value of the `id` field on every instance of the following type of object in a model must be unique: **Model**, **FunctionDefinition**, **CompartmentType**, **Compartment**, **SpeciesType**, **Species**, **Reaction**, **SpeciesReference**, **ModifierSpeciesReference**, **Event**, and model-wide **Parameter**s. Note that **UnitDefinition** and parameters defined inside a reaction are treated separately.

10302. The value of the `id` field of every **UnitDefinition** must be unique across the set of all **UnitDefinition**s in the entire model.

10303. The value of the `id` field of each parameter defined locally within a **KineticLaw** must be unique across the set of all such parameter definitions in that **KineticLaw**.

10304. The value of the `variable` field in all **AssignmentRule** and **RateRule** definitions must be unique across the set of all such rule definitions in a model.

10305. In each **Event**, the value of the `variable` field within every **EventAssignment** definition must be unique across the set of all **EventAssignment**s within that **Event**.

10306. An identifier used as the value of `variable` in an **EventAssignment** cannot also appear as the value of `variable` in an **AssignmentRule**.

10307. Every `metaid` field value must be unique across the set of all `metaid` values in a model.

10308. The value of a `sboTerm` attribute must have the data type `SBOTerm`, which is a string consisting of the characters 'S', 'B', 'O', ':', followed by exactly seven digits.

10309. The syntax of `metaid` field values must conform to the syntax of the XML type `ID`.

10310. The syntax of `id` field values must conform to the syntax of the SBML type `SId`.

General Annotation validation

10401. Every top-level element within an `annotation` element must have a namespace declared.

10402. There cannot be more than one top-level element using a given namespace inside a given `annotation` element.

10403. Top-level elements within an `annotation` element cannot use any SBML namespace, whether explicitly (by declaring the namespace to be one of the URIs "`http://www.sbml.org/sbml/level1`", "`http://www.sbml.org/sbml/level2`", or "`http://www.sbml.org/sbml/level2/version2`"), or implicitly (by failing to declare any namespace).

General Unit validation

10501. The units of the expressions used as arguments to a function call must match the units expected for the arguments of that function.

10511. When the `variable` in an **AssignmentRule** refers to a **Compartment**, the units of the rule's right-hand side must be consistent with the units of that compartment's size.

10512. When the `variable` in an **AssignmentRule** refers to a **Species**, the units of the rule's right-hand side must be consistent with the units of the species' quantity.

10513. When the `variable` in an **AssignmentRule** refers to a **Parameter**, the units of the rule's right-hand side must be consistent with the units declared for that parameter.

10521. When the `variable` in an **InitialAssignment** refers to a **Compartment**, the units of the **InitialAssignment**'s `math` expression must be consistent with the units of that compartment's size.

10522. When the `variable` in an **InitialAssignment** refers to a **Species**, the units of the **InitialAssignment**'s `math` expression must be consistent with the units of that species' quantity.

10523. When the `variable` in an **InitialAssignment** refers to a **Parameter**, the units of the **InitialAssignment**'s `math` expression must be consistent with the units declared for that parameter.

10531. When the `variable` in a **RateRule** definition refers to a **Compartment**, the units of the rule's right-hand side must be of the form *x per time*, where *x* is either the `units` in that **Compartment** definition, or (in the absence of explicit units declared for the compartment size) the default units for that compartment, and *time* refers to the units of time for the model.

10532. When the `variable` in a **RateRule** definition refers to a **Species**, the units of the rule's right-hand side must be of the form *x per time*, where *x* is the units of that species' quantity, and *time* refers to the units of time for the model.

10533. When the `variable` in a **RateRule** definition refers to a **Parameter**, the units of the rule's right-hand side must be of the form *x per time*, where *x* is the `units` in that **Parameter** definition, and *time* refers to the units of time for the model.

10541. The units of the `math` formula in a **KineticLaw** definition must be the equivalent of *substance per time*.

10551. When a value for `delay` is given in a **Event** definition, the units of the delay formula must correspond to either the value of `timeUnits` in the **Event** or (if no `timeUnits` are given), the model's default units of time.

General Model validation

10601. The system of equations created from an SBML model must not be overdetermined.

SBML container validation

20101. The `sbml` container element must declare the XML Namespace for SBML, and this declaration must be consistent with the values of the `level` and `version` attributes on the `sbml` element.

20102. The `sbml` container element must declare the SBML Level using the attribute `level`, and this declaration must be consistent with the XML Namespace declared for the `sbml` element.

20103. The `sbml` container element must declare the SBML Version using the attribute `version`, and this declaration must be consistent with the XML Namespace declared for the `sbml` element.

Model validation

20201. An SBML document must contain a **Model** definition. .

20202. The order of subelements within a **Model** object instance must be the following (where any one may be optional): `listOfFunctionDefinitions`, `listOfUnitDefinitions`, `listOfCompartmentTypes`, `listOfSpeciesTypes`, `listOfCompartments`, `listOfSpecies`, `listOfParameters`, `listOfInitialAssignments`, `listOfRules`, `listOfConstraints`, `listOfReactions`, `listOfEvents`.

20203. The `listOf___` containers in a **Model** are optional, but if present, the lists cannot be empty. Specifically, if any of the following are present in a **Model**, they must not be empty: `listOfFunctionDefinitions`, `listOfUnitDefinitions`, `listOfCompartmentTypes`, `listOfSpeciesTypes`, `listOfCompartments`, `listOfSpecies`, `listOfParameters`, `listOfInitialAssignments`, `listOfRules`, `listOfConstraints`, `listOfReactions` and `listOfEvents`.

20204. If a model defines any **Species**, then the model must also define at least one **Compartment**. This is an implication of the fact that the `compartment` field on **Species** is not optional.

FunctionDefinition validation

20301. The top-level element within `math` in a **FunctionDefinition** must be `lambda`.

20302. Inside the `lambda` of a **FunctionDefinition**, if a `ci` element is the first element within a MathML `apply`, then the `ci`'s value can only be chosen from the set of identifiers of other SBML **FunctionDefinition**s defined prior to that point in the SBML model. In other words, forward references to user-functions are not permitted.

20303. Inside the `lambda` of a **FunctionDefinition**, the identifier of that **FunctionDefinition** cannot appear as the value of a `ci` element. SBML functions are not permitted to be recursive.

20304. Inside the `lambda` of a **FunctionDefinition**, if a `ci` element is not the first element within a MathML `apply`, then the `ci`'s value can only be the value of a `bvar` element declared in that `lambda`. In other words, all model entities referenced inside a function definition must be passed arguments to that function.

20305. The value type returned by a **FunctionDefinition**'s `lambda` must be either boolean or numeric.

Unit and UnitDefinition validation

20401. The value of the `id` field in a **UnitDefinition** must not be identical to any unit predefined in SBML. That is, the identifier must not be the same as a value from the `UnitKind` enumeration (i.e., "`ampere`" "`gram`" "`katal`" "`metre`" "`second`" "`watt`" "`becquerel`" "`gray`" "`kelvin`" "`mole`" "`siemens`" "`weber`" "`candela`" "`henry`" "`kilogram`" "`newton`" "`sievert`" "`coulomb`" "`hertz`" "`litre`" "`ohm`" "`steradian`" "`dimensionless`" "`item`" "`lumen`" "`pascal`" "`tesla`" "`farad`" "`joule`" "`lux`" "`radian`" "`volt`".)

20402. Redefinitions of the built-in unit `substance` must be based on the units `mole`, `item`, `gram`, `kilogram`, or `dimensionless`. More formally, a **UnitDefinition** for `substance` must simplify to a single **Unit** whose `kind` field has a value of "`mole`", "`item`", "`gram`", "`kilogram`", or `dimensionless`, and whose `exponent` field has a value of "`1`".

20403. Redefinitions of the built-in unit `length` must be based on the unit `metre` or `dimensionless`. More formally, a **UnitDefinition** for `length` must simplify to a single **Unit** in which either (a) the `kind` field has a value of "`metre`" and the `exponent` field has a value of "`1`", or (b) the `kind` field has a value of "`dimensionless`" with any `exponent` value.

20404. Redefinitions of the built-in unit `area` must be based on squared `metre`s or `dimensionless`. More formally, a **UnitDefinition** for `area` must simplify to a single **Unit** in which either (a) the `kind` field has a value of "`metre`" and the `exponent` field has a value of "`2`", or (b) the `kind` field has a value of "`dimensionless`" with any `exponent` value.

20405. Redefinitions of the built-in unit `time` must be based on `second`. More formally, a **UnitDefinition** for `time` must simplify to a single **Unit** in which either (a) the `kind` field has a value of "`second`" and the `exponent` field has a value of "`1`", or (b) the `kind` field has a value of "`dimensionless`" with any `exponent` value.

20406. Redefinitions of the built-in unit `volume` must be based on `litre`, `metre` or `dimensionless`. More formally, a **UnitDefinition** for `volume` must simplify to a single **Unit** in which the `kind` field value is either "`litre`", "`metre`", or "`dimensionless`". Additional constraints apply if the kind is "`litre`" or "`metre`".

20407. If a **UnitDefinition** for `volume` simplifies to a **Unit** in which the `kind` field value is "`litre`", then its `exponent` field value must be "`1`".

20408. If a **UnitDefinition** for `volume` simplifies to a **Unit** in which the `kind` field value is "`metre`", then its `exponent` field value must be "`3`".

20409. The `listOfUnits` container in a **UnitDefinition** cannot be empty.

20410. The value of the `kind` field of a **Unit** can only be one of the predefined units enumerated by `UnitKind`; that is, the SBML unit system is not hierarchical and user-defined units cannot be defined using other user-defined units.

20411. The `offset` field on **Unit** previously available in SBML Level 2 Version 1, has been removed as of SBML Level 2 Version 2.

20412. The predefined unit "`Celsius`", previously available in SBML Level 1 and Level 2 Version 1, has been removed as of SBML Level 2 Version 2.

Compartment validation

20501. The size of a **Compartment** must not be set if the compartment's `spatialDimensions` field has value `0`.

20502. If a **Compartment** definition has a `spatialDimensions` value of "`0`", then its `units` field must not be set. If the compartment has no dimensions, then no units can be associated with a non-existent size.

20503. If a **Compartment** definition has a `spatialDimensions` value of "`0`", then its `constant` field value must either default to or be set to "`true`". If the compartment has no dimensions, then its size can never change.

20504. The `outside` field value of a **Compartment** must be the identifier of another **Compartment** defined in the model.

20505. A **Compartment** may not enclose itself through a chain of references involving the `outside` field. This means that a compartment cannot have its own identifier as the value of `outside`, nor can it point to another compartment whose `outside` field points directly or indirectly to the compartment.

20506. The `outside` field value of a **Compartment** cannot be a compartment whose `spatialDimensions` value is "`0`", unless both compartments have `spatialDimensions`="`0`". Simply put, a zero-dimensional compartment cannot enclose compartments that have anything other than zero dimensions themselves.

20507. The value of the `units` field on a **Compartment** having `spatialDimensions` of "`1`" must be either "`length`", "`metre`", "`dimensionless`", or the identifier of a **UnitDefinition** based on either `metre` (with `exponent` equal to "`1`") or `dimensionless`.

20508. The value of the `units` field on a **Compartment** having `spatialDimensions` of "`2`" must be either "`area`", "`dimensionless`", or the identifier of a **UnitDefinition** based on either `metre` (with `exponent` equal to "`2`") or `dimensionless`.

20509. The value of the `units` field on a **Compartment** having `spatialDimensions` of "3" must be either "`volume`", "`litre`", or the identifier of a **UnitDefinition** based on either `litre`, `metre` (with `exponent` equal to "3"), or `dimensionless`.

20510. If the `compartmentType` field is given a value in a **Compartment** definition, it must contain the identifier of an existing **CompartmentType**.

Species validation

20601. The value of `compartment` in a **Species** definition must be the identifier of an existing **Compartment** defined in the model.

20602. If a **Species** definition sets `hasOnlySubstanceUnits` to "`true`", then it must not have a value for `spatialSizeUnits`.

20603. A **Species** definition must not set `spatialSizeUnits` if the **Compartment** in which it is located has a `spatialDimensions` value of "0".

20604. If a **Species** located in a **Compartment** whose `spatialDimensions` is set to "0", then that **Species** definition cannot set `initialConcentration`.

20605. If a **Species** is located in a **Compartment** whose `spatialDimensions` has value "1", then that **Species** definition can only set `spatialSizeUnits` to a value of "`length`", "`metre`", "`dimensionless`", or the identifier of a **UnitDefinition** derived from `metre` (with an `exponent` value of "1") or "`dimensionless`".

20606. If a **Species** is located in a **Compartment** whose `spatialDimensions` has value "2", then that **Species** definition can only set `spatialSizeUnits` to a value of "`area`", "`dimensionless`", or the identifier of a **UnitDefinition** derived from either `metre` (with an `exponent` value of "2") or "`dimensionless`".

20607. If a **Species** is located in a **Compartment** whose `spatialDimensions` has value "3", then that **Species** definition can only set `spatialSizeUnits` to a value of "`volume`", "`litre`", "`dimensionless`", or the identifier of a **UnitDefinition** derived from either `litre`, `metre` (with an `exponent` value of "3") or `dimensionless`.

20608. The value of a **Species**'s `substanceUnits` field can only be one of the following: "`substance`", "`mole`", "`item`", "`gram`", "`kilogram`", "`dimensionless`", or the identifier of a **UnitDefinition** derived from "`mole`" (with an `exponent` of "1"), "`item`" (with an `exponent` of "1"), "`gram`" (with an `exponent` of "1"), "`kilogram`" (with an `exponent` of "1"), or "`dimensionless`".

20609. A **Species** cannot set values for both `initialConcentration` and `initialAmount` because they are mutually exclusive.

20610. A **Species**' quantity cannot be determined simultaneously by both reactions and rules. More formally, if the identifier of a **Species** definition having `boundaryCondition`="`false`" and `constant`="`false`" is referenced by a **SpeciesReference** anywhere in a model, then this identifier cannot also appear as the value of a `variable` in an **AssignmentRule** or a **RateRule**.

20611. A **Species** having `boundaryCondition`="`false`" cannot appear as a reactant or product in any reaction if that **Species** also has `constant`="`true`".

20612. The value of `speciesType` in a **Species** definition must be the identifier of an existing **SpeciesType**.

20613. There cannot be more than one species of a given **SpeciesType** in the same compartment of a model. More formally, for any given compartment, there cannot be more than one **Species** definition in which both of the following hold simultaneously: (i) the **Species**' `compartment` value is set to that compartment's identifier and (ii) the **Species**' `speciesType` is set the same value as the `speciesType` of another **Species** that also sets its `compartment` to that compartment identifier.

Parameter validation

20701. The `units` in a **Parameter** definition must be a value chosen from among the following: a value from the `UnitKind` enumeration (e.g., "`litre`", "`mole`", "`metre`", etc.), a built-in unit (e.g., "`substance`", "`time`", etc.), or the identifier of a **UnitDefinition** in the model.

InitialAssignment validation

20801. The value of `symbol` in an **InitialAssignment** definition must be the identifier of an existing **Compartment**, **Species**, or **Parameter** defined in the model.

20802. A given identifier cannot appear as the value of more than one `symbol` field across the set of **InitialAssignment**s in a model.

20803. The value of a `symbol` field in any **InitialAssignment** definition cannot also appear as the value of a `variable` field in an **AssignmentRule**.

AssignmentRule and RateRule validation

20901. The value of an **AssignmentRule**'s `variable` must be the identifier of an existing **Compartment**, **Species**, or globally-defined **Parameter**.

20902. The value of a **RateRule**'s `variable` must be the identifier of an existing **Compartment**, **Species**, or globally-defined **Parameter**.

20903. Any **Compartment**, **Species** or **Parameter** whose identifier is the value of a `variable` field in an **AssignmentRule**, must have a value of "`false`" for `constant`.

20904. Any **Compartment**, **Species** or **Parameter** whose identifier is the value of a `variable` field in an **RateRule**, must have a value of "`false`" for `constant`.

20905. A given identifier cannot appear as the value of more than one `variable` field across the combined set of **AssignmentRule**s and **RateRule**s in a model.

20906. There must not be circular dependencies in the combined set of **InitialAssignment**, **AssignmentRule** and **KineticLaw** definitions in a model. Each of these constructs has the effect of assigning a value to an identifier (i.e., the identifier given in the field `symbol` in **InitialAssignment**, the field `variable` in **AssignmentRule**, and the field `id` on the **KineticLaw**'s enclosing **Reaction**). Each of these constructs computes the value using a mathematical formula. The formula for a given identifier cannot make reference to a second identifier whose own definition depends directly or indirectly on the first identifier.

Constraint validation

21001. A **Constraint** `math` expression must evaluate to a value of type `boolean`.

21002. The order of subelements within **Constraint** must be the following: `math`, `message`. The `message` element is optional, but if present, must follow the `math` element.

Reaction validation

21101. A **Reaction** definition must contain at least one **SpeciesReference**, either in its `listOfReactants` or its `listOfProducts`. A reaction without any reactant or product species is not permitted, regardless of whether the reaction has any modifier species.

21102. The order of subelements within **Reaction** must be the following (where every one is optional): `listOfReactants`, `listOfProducts`, `listOfModifiers`, `kineticLaw`.

21103. The following containers are all optional in a **Reaction**, but if any present is, it must not be empty: `listOfReactants`, `listOfProducts`, `listOfModifiers`, `kineticLaw`.

21104. The list of reactants (`listOfReactants`) and list of products (`listOfProducts`) in a **Reaction** can only contain `speciesReference` elements.

21105. The list of modifiers (`listOfModifiers`) in a **Reaction** can only contain `modifierSpeciesReference` elements.

SpeciesReference and ModifierSpeciesReference validation

21111. The value of a **SpeciesReference** `species` field must be the identifier of an existing **Species** in the model.

21112. The value of a **SpeciesReference**'s `species` field must not be the identifier of a **Species** having both a `constant` field value of "`true`" and a `boundaryCondition` field value of "`false`".

21113. A **SpeciesReference** must not have a value for both `stoichiometry` and `stoichiometryMath`; they are mutually exclusive.

KineticLaw validation

21121. All species referenced in the **KineticLaw** formula of a given reaction must first be declared using **SpeciesReference** or **ModifierSpeciesReference**. More formally, if a **Species** identifier appears in a `ci` element of a **Reaction**'s **KineticLaw** formula, that same identifier must also appear in at least one **SpeciesReference** or **ModifierSpeciesReference** in the **Reaction** definition.

21122. The order of subelements within **KineticLaw** must be the following: `math`, `listOfParameters`. The `listOfParameters` is optional, but if present, must follow `math`.

21123. If present, the `listOfParameters` in a **KineticLaw** must not be an empty list.

21124. The `constant` field on a **Parameter** local to a **KineticLaw** cannot have a value other than "`true`". The values of parameters local to **KineticLaw** definitions cannot be changed, and therefore they are always constant.

21125. The `substanceUnits` field on **KineticLaw**, previously available in SBML Level 1 and Level 2 Version 1, has been removed as of SBML Level 2 Version 2. In SBML Level 2 Version 2, the substance units of a reaction rate expression are those of the global "`substance`" units of the model.

21126. The `timeUnits` field on **KineticLaw**, previously available in SBML Level 1 and Level 2 Version 1, has been removed as of SBML Level 2 Version 2. In SBML Level 2 Version 2, the time units of a reaction rate expression are those of the global "`time`" units of the model.

StoichiometryMath validation

21131. All species referenced in the **StoichiometryMath** formula of a given reaction must first be declared using **SpeciesReference** or **ModifierSpeciesReference**. More formally, if a **Species** identifier appears in a `ci` element of a **Reaction**'s **StoichiometryMath** formula, that same identifier must also appear in at least one **SpeciesReference** or **ModifierSpeciesReference** in the **Reaction** definition.

Event validation

21201. An **Event** object must have a `trigger`.

21202. An **Event** `trigger` expression must evaluate to a value of type `boolean`.

21203. An **Event** object must have at least one **EventAssignment** object in its `listOfEventAssignments`.

21204. The value of an **Event**'s `timeUnits` must be "`time`", "`second`", "`dimensionless`", or the identifier of a **UnitDefinition** derived from either `second` (with an `exponent` value of "`1`") or "`dimensionless`".

21205. The order of subelements within an **Event** object instance must be the following: `trigger`, `delay`, `listOfEventAssignments`. The `delay` element is optional, but if present, must follow `trigger`.

EventAssignment validation

21211. The value of `variable` in an **EventAssignment** can only be the identifier of a **Compartment**, **Species**, or model-wide **Parameter** definition.

21212. Any **Compartment**, **Species** or **Parameter** definition whose identifier is used as the value of `variable` in an **EventAssignment** must have a value of "`false`" for its `constant` field.

Integrate a Units Checking and Translation System The beta version of *libSBML* 3.0 available at the end of the phase of BioSPICE funding contained a unit verification system. This system operates by converting all units to the base units (which are the SI units), and then verifying that the resulting overall mathematical system is consistent. The beta release of *libSBML* has been available since April, 2006, via the SourceForge.net site for SBML.

Provide an API for SBML Annotations The question arose about what an API for SBML annotations might look like. We decided to provide a DOM-like approach (where "DOM" is domain object model, a standard XML interface approach available in many XML parser libraries). This is not as specific to SBML as might be desired; on the other hand, we could not find a better approach that balanced SBML specificity against generality such that any kind of annotation (even future ones unpredictable today) could be handled.

In SBML Level 2 Version 2, the SBML annotations were restricted to one top- level element per annotation. This restriction was put in place to both make it easier to process existing annotations and to allow programs to add annotations without clobbering existing annotation data. *libSBML* 3.0 complements the SBML L2V2 annotation scheme by converting annotation data into an XML Document Object Model (DOM) data structure. The *libSBML* XML DOM provides an query API to search for elements with particular names and/or namespaces so that programmers can quickly access and manipulate only those annotation elements which are pertinent to their applications.

Provide Additional "Convenience" APIs LibSBML 3.0 provides several APIs for working with SBML's UnitDefinitions, XML DOMs, and extending the *libSBML* parser. UnitDefinitions can be queried for equivalence classes (e.g., isVariantOfMass()). The XML DOM API provides a convenient way to store, process, and round-trip any XML that *libSBML* does not understand. Finally, programmers may register additional handlers with the parser to create custom data structures for XML content that would be too large or inconvenient to store and manipulate as an XML DOM. This is how the *libSBML* Layout extension has been implemented and plugged-into libSBML.

Implement Support for RDF and Metadata in SBML RDF Metadata support is still in progress for *libSBML* 3.0. The SBML L2V2 specification recommends a particular form and structure for annotations. This scheme establishes a canonical form for RDF metadata annotations as they pertain to SBML (in the same way that SBML restricts XML as it pertains to biochemical reaction networks). Currently, *libSBML* defaults to XML DOM construction for RDF metadata. While this default implementation is operable, we believe we can a better experience for programmers. We are currently reviewing API designs to query and manipulate the L2V2 structure in a more straightforward manner.

Support "Within-SBML" Translations Many software tools supporting SBML do not support all features of the language. However, in the case of some of these features, it is actually possible to "translate" the model to an equivalent model that uses different features within SBML. The development of *libSBML* has considered the need to perform such translations and has focussed on providing a framework whereby such translation is possible.

One area where there is little support is the area of units. The majority of tools cannot deal with units on quantities in a model, and thus any models where the user has used either a mix of units or less common units cannot be simulated correctly by many tools. The development of *libSBML* has focussed on the validation of the units used in a model, both in terms of mathematical

manipulation (for example checking that formulae used do not add quantities of differing units) and the appropriate overall units resulting from a formula (for example checking that a formula assigning a value to a compartment volume has units of volume). However, the framework developed to provide this validation can be used to change units within the model and thus translate a model with a variety of units into a model with consistent units.

Another feature not supported by many tools is the user defined function. Again, focussing on validation, *libSBML* provides a framework that could be used to replace user defined functions with their intended functionality directly into any math element using the function.

Our progress to date in this area has been to rework the internal *libSBML* implementation to support these kinds of operations. While *libSBML* does not as yet provide a direct API for performing these translations, the framework has been rigorously constructed to facility their addition.

Support Translating SBML Level 2 down to SBML Level 1 Some models written in SBML Level 2 could be written in SBML Level 1 and thus made available to those tools only supporting SBML L1. *libSBML* provides the facility to perform this translation where appropriate.

libSBML stores an SBML model as an object with associated values. While obviously the level and version of SBML are as part of this object, the individual values are not level specific. For example the `initialAmount` or `initialConcentration` value of a species, which are mutually exclusive, have only one associated variable and the intention of the user, concentration or amount, is recorded by means of flags. When *libSBML* outputs a model it outputs the attributes and associated values appropriate for the level and version of SBML. This allows a user to change the level of a model after it has been read in via the `setLevel()` function and the on outputting the model written out will be of the desired level. The user need make no other alteration to the model.

However, there is a subset of models written in SBML Level 2 that cannot be written as valid SBML Level 1 models; for example, models that use components such as events. In order to prevent these models being written out in SBML Level 1 with missing information, *libSBML* uses a instance of the validator class (used for the validation of SBML within *libSBML*) which applies a number of rules to the model to determine whether it is compatible with Level 1. This class object is called when *libSBML* detects a request for a change of level from 2 to 1 and if the model fails the compatibility tests this is reported to the user and the level of the model object in *libSBML* is not changed.

Support SBML Level 2 Version 2, Expected to Be Introduced in 2005 LibSBML 3.0 beta supports SBML Level 2 Version 2. The beta version was made available in April 2006, corresponding to a draft of SBML Level 2 Version 2 available at that time. The final Version 2 specification was not issued until September 2006, and the final *libSBML* 3.0 supports the final Version 2 specification.

3.2.2 Support of SBML Use and Evolution

Coordination of Annotations In the last year of the BioSPICE project, we surveyed the BioSPICE user community and the broader SBML community about the annotations in use, and reported this on a web page on the sbml.org project website. We summarize the results of this survey below.

- *JigCell* (`http://jigcell.biol.vt.edu`). This set of annotations is used by the JigCell software system to add information about rate laws and conservation laws in a model.

- *JDesigner* (`http://www.sys-bio.org/software/jdesigner.htm`) JDesigner is a Win32 application which allows one to draw a biochemical network and export the network in the form of SBML. JDesigner has an SBW interface that allows it to be called from other SBW compliant modules, for example Python. In addition, JDesigner has the ability to use Jarnac as a simulation server (via SBW) thus allowing models to be run from within JDesigner. This annotation also includes provision for named rate laws (see UPenn BioCharon annotations).

- *BioCharon* (`http://www.cis.upenn.edu/biocomp/Hybrid`) We are using a list of rate laws that were part of the SBML Level 1 specification, but we add reaction types that appear useful as we try to annotate various models. The primary purpose is to facilitate the automated construction of hybrid approximations to biochemical network models described in SBML files. However, the issue of easily recognizing common reaction law types is also faced by other software that uses SBML. The annotations include the terms *ConstantLaw*, *MassActionLaw*, *MichaelisMentenLaw*, *HillLaw*, and *MichaelisMentenReversibleLaw*.

- *Karyote* (`http://biodynamics.indiana.edu/cyber_cell/`) The dynamics of a living cell are modeled using a comprehensive reaction, transport, genomic approach. The calibration of the model and the difficulty arising from the incompleteness of our (and any) model of such a complex system are addressed via an information theory approach. Species annotations contain internal ids of the species in the Karyote database.

- *SBML Layout Extension* (`http://projects.eml.org/bcb/sbml/`) At the 9th SBML Forum in Heidelberg, Oct. 2005, this was voted as the recommended approach for storing diagrams in SBML. The recommendation is to use annotations to store the information in SBML Level 2. In SBML Level 3, it will become a standard package; when this happens, the tags will no longer have to be placed inside annotations but instead will be graduated to top-level SBML status.

- *BioUML* (`http://www.biouml.org/index.shtml`) BioUML is Java framework for systems biology. It spans the comprehensive range of capabilities including access to databases with experimental data, tools for formalized description of biological systems structure and functioning, as well as tools for their visualization and simulations. Developed a diagram layout extension for use with the BioUML Workbench.

- *BioModels Database* (`http://biomodels.net/biomodels`) Biomodels is a database of quantitative models in biology. The current format to feed the database and to export the results is SBML Level 2 Version 1. Annotations used are "submitter" (dc:contributor plus vCard), "creator" (dc:creator plus vCard) and links to other resources (dc:relation, cf. proposal of Andrew Finney and Nicolas Le Novre 2004)

- *MOMA* (`http://arep.med.harvard.edu/fluxns`) Metabolic flux analysis requires the ability to constrain reaction fluxes so that a feasible space can be obtained. The current SBML specification does not have a natural place for this information, so we add it to annotation. In the future, we expect this will be replaced by a general-purpose constraints specification.

BioPAX and SBML During the BioSPICE phase of *libSBML* funding, we initiated an effort to define a proposal for how to connect SBML and BioPAX. We reached the point of defining three general approaches. The efforts then stalled because there were no consumers of BioPAX-format pathway representations at the time, and therefore it was impossible to assertain which approach would be more suitable for real software tools. The BioPAX-SBML effort was put on hold until

such time as there exist software systems that can import BioPAX-formatted pathway data. This goal of the original project remains unmet.

4 Summary

4.1 Sum of Squares (SOS) Framework and SOSTOOLS

The overall long-term objective of our research has been to develop mathematical and software infrastructure in support of post-genomics research in systems biology, including a deeper understanding of the organizational principles of biological networks. A distinguishing theme of this work is its focus on scalable methods of robustness and model (in)validation with data, as opposed to relying purely on simulation. In computability terms, if simulation is viewed as a way to attack the NP hard side of biological problems, our approach attacks the coNP side. Much of the success of reductionist biology has depended on creative individuals who draw biologically meaningful inferences from data and computation using small scale and informal reasoning. This type of inference was critical because the reductionist research program itself offered no systematic tools to deal with complexity, only with the component parts. Far from being dispensed with, this reasoning process and its biological content must be both formalized and made rigorous, systematic, and scalable as well, and ultimately teachable. This requires the development of new mathematics as well as algorithms and software.

A central goal of modeling and simulation is to connect molecular mechanisms to network function to questions of biomedical relevance. Unfortunately, many of the most critical questions involve events which are extremely rare at the individual cell level where the mechanisms act yet catastrophic to the organism. Thus simulation methods that may be adequate for studying generic or typical behavior are entirely inadequate to explore such worst-case scenarios, which with conventional methods are computational intractable. We have extended the best-practice tools and algorithms for robustness analysis that have become standards in engineering to models of biological relevance, which are typically nonlinear, hybrid, uncertain, and stochastic [34]. This includes SOSTOOLS, an open source software implementation [92].

4.2 SBML

Computational modeling is becoming crucial for making sense of the vast quantities of complex experimental data that are now being collected. The systems biology community needs agreed-upon information standards if models are to be shared, evaluated and developed cooperatively. The Systems Biology Markup Language (SBML) is an XML-based format for representing computational models in a way that can be used by different software systems to communicate and exchange those models. It is supported today by over 100 software tools worldwide and a vibrant community of modelers and software authors. A variety of resources are available for working with SBML; there is also an Internet MIME type defined for SBML [56] and a new public database of models based around SBML [8].

In support of SBML and its community, we continue to develop and make available software infrastructure, including programming libraries, conversion utilities, interface packages for commonly-used software environments, and easy-to-access Internet-accessible online tools. All of our software development follows the open-source tradition to maximize the accessibility and utility of the products.

We note with satisfaction that the open-source model of software development has been yielding dividends for *libSBML*. The user community has contributed not only several bug fixes, but new

code as well. These include: support for the Expat parser library, a full Perl API, a full Lisp API, and an extension to support the use of a provisional SBML standard for storing model diagrams.

The *libSBML* open-source license allows it to be incorporated freely into other programs in whole or part. Several simulator programs and projects developed in academia already make use of *libSBML* to support both SBML import and export. It is worth mentioning that since *libSBML* is distributed under the terms of the Lesser GNU Public License (LGPL) it may also be used without restriction in commercial applications [39]. We currently know of two commercial software applications using *libSBML*.

References

[1] B. Alberts et al., *Molecular Biology of the Cell*, Garland Publishing, New York, 2002.

[2] A. Arkin, J. Ross, and H. McAdams, *Stochastic kinetic analysis of the developmental pathway bifurcation in phage λ-infected* escehrichia coli *cells*, Genetics, 149 (1998), pp. 1633–1648.

[3] R. Ausbrooks, S. Buswell, S. Dalmas, S. Devitt, A. Diaz, R. Hunter, B. Smith, N. Soiffer, R. Sutor, and S. Watt, *Mathematical markup language (MathML) version 2.0: W3C Recommendation 21 February 2001*, 2001. Available on the Internet at http://www.w3.org/TR/2001/REC-MathML2-20010221.

[4] G. D. Bader, E. Brauner, M. P. Cary, R. Goldberg, C. Hogue, P. Karp, T. Klein, J. Luciano, D. Marks, N. Maltsev, E. Marland, E. Neumann, S. Paley, J. Rick, A. Regev, A. Rzhetsky, C. Sander, V. Schachter, I. Shah, and J. Zucker, *BioPAX–Biological Pathways Exchange language. Level 1, Version 1.4 documentation.* Available on the Internet at `http://www.biopax.org`, 2005.

[5] M. Barahona, A. C. Doherty, M. Sznaier, H. Mabuchi, and J. C. Doyle, *Finite horizon model reduction and the appearance of dissipation in hamiltonian systems*, in Proc. of IEEE Conference on Decision and Control, 2002.

[6] K. Beck, *Test Driven Development: By Example*, Addison-Wesley Professional, 2002.

[7] D. Bertsimas and I. Popescu, *Optimal inequalities in probability: A convex optimization approach.* INSEAD working paper, available at `http://www.insead.edu/facultyresearch/tm/popescu/`, 1999–2001.

[8] BioModels Team, *The BioModels database.* Available via the World Wide Web at `http://www.ebi.ac.uk/biomodels`., 2005.

[9] BioPAX Working Group, *Biopax: Biological pathways exchange.* Available on the Internet at `http://www.biopax.org`, 2005.

[10] K. M. Bobba, B. Bamieh, and J. C. Doyle, *Robustness and Navier-Stokes Equations*, in In Proceedings of the 41st IEEE Conference on Decision and Control, 2002.

[11] K. M. Bobba, J. C. Doyle, and M. Gharib, *A reynolds number independent model for turbulence in couette flow*, in Proc. of IUTAM Symposium on Reynolds Number Scaling in Turbulent Flows, 2002.

[12] ——, *Stochastic input-output measures for transition to turbulence*, in Proc. of AIAA Aerospace Sciences Meeting, AIAA Paper No. 2003-0786, 2003.

[13] ——, *Techniques for simplifying multiscale, linear fluid dynamics problems*, in Proc. of SIAM Conference on Applied Linear Algebra, 2003.

[14] J. Bochnak, M. Coste, and M.-F. Roy, *Real Algebraic Geometry*, Springer, 1998.

[15] J. Bosak and T. Bray, *XML and the second-generation web*, Scientific American, (1999).

[16] J. M. Bower and H. Bolouri, *Computational Modeling of Genetic and Biochemical Networks*, MIT Press, Cambridge, Mass., 2001.

[17] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory*, Society for Industrial and Applied Mathematics (SIAM), 1994.

[18] T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler, *Extensible markup language (XML) 1.0 (second edition), W3C recommendation 6-October-2000*, tech. rep., 2000. Available on the Internet at `http://www.w3.org/TR/1998/REC-xml-19980210`.

[19] S. L. Campbell, *Singular Systems of Differential Equations*, Pitman, Boston, MA, 1980.

[20] J. M. Carlson and J. C. Doyle, *Highly Optimized Tolerance: a mechanism for power laws in designed systems*, Physical Review E, 60 (1999), pp. 1412–1428.

[21] ——, *Highly Optimized Tolerance: Robustness and design in complex systems*, Physical Review Letters, 84 (2000), pp. 2529–2532.

[22] J. M. Carlson and J. C. Doyle, *Complexity and robustness*, in Proc. Natl. Acad. Sci. USA, 99, Suppl. 1, 2002, pp. 2538–2545.

[23] M. D. Choi, T. Y. Lam, and B. Reznick, *Sum of squares of real polynomials*, Proceedings of Symposia in Pure Mathematics, 58 (1995), pp. 103–126.

[24] M. Csete and J. Doyle, *Bowties, metabolism, and disease*, Trends in Biotechnology, 22 (2004), pp. 446–450.

[25] M. E. Csete and J. C. Doyle, *Reverse engineering of biological complexity*, Science, 295 (2002), pp. 1664–1669.

[26] L. Dai, *Singular Control Systems*, Springer-Verlag, New York, 1989.

[27] C. A. Desoer and M. Vidyasagar, *Feedback Systems: Input-Output Properties*, Academic Press, New York, 1975.

[28] A. C. Doherty, P. A. Parrilo, and F. M. Spedalieri, *Distinguishing separable and entangled states*, Physical Review Letters, 88 (2002).

[29] J. C. Doyle and J. M. Carlson, *Power laws, Highly Optimized Tolerance and generalized source coding*, Physical Review Letters, 84 (2000), pp. 5656–5659.

[30] G. Dullerud and R. Smith, *A nonlinear functional approach to LFT model validation*, Systems and Control Letters, 47 (2002), pp. 1–11.

[31] H. El-Samad, M. Khammash, H. Kurata, and J. Doyle, *Robustness analysis of the heat shock response in* e. coli, in Proceedings of the American Control Conference, 2002, pp. 1742–1747.

[32] H. El-Samad, H. Kurata, J. Doyle, C. Gross, and M. Khammash, *Surviving heat shock: Control strategies for robustness and performance*, Proc. Natl. Acad. Sci. USA, 102 (2005), pp. 2736–2741.

[33] H. El-Samad, A. Papachristodoulou, S. Prajna, J. C. Doyle, and M. H. Khammash, *Advanced methods and algorithms for biological networks analysis*, Proceedings of the IEEE, (2006). To appear.

[34] H. El-Samad, S. Prajna, A. Papachristodoulou, M. H. Khammash, and J. C. Doyle, *Model validation and robustness analysis of the bacterial heat shock response using SOSTOOLS*, in Proceedings IEEE Conference on Decision and Control, 2003.

[35] R. Feeley, P. Seiler, A. Packard, and M. Frenklach, *Consistency of a reaction data set*, Journal of Physical Chemistry A, 108 (2004), pp. 9573–9583.

[36] E. Feron, P. Apkarian, and P. Gahinet, *Analysis and synthesis of robust control systems via parameter-dependent Lyapunov functions*, IEEE Transactions on Automatic Control, 41 (1996), pp. 1041–1046.

[37] A. Finney, M. Hucka, and H. Bolouri, *Systems Biology Markup Language (SBML) Level 2: Structures and facilities for model definitions*. Available via the World Wide Web at `http://sbml.org/documents/`, 2002.

[38] A. Finney, M. Hucka, H. Sauro, H. Bolouri, A. Funahashi, B. Bornstein, B. Kovitz, J. Matthews, B. E. Shapiro, S. Keating, J. Doyle, and H. Kitano, *The systems biology workbench (sbw) version 1.0: Framework and modules*, Jan, 2003 2003.

[39] Free Software Foundation, *The GNU lesser general public license*. Available via the World Wide Web at `http://www.fsf.org/licenses/licenses.html`, 1999.

[40] M. Frenklach, A. Packard, P. Seiler, and R. Feeley, *Collaborative data processing in developing predictive models of complex reaction systems*, International Journal of Chemical Kinetics, 36 (2004), pp. 57–66.

[41] K. Gatermann and P. A. Parrilo, *Symmetry groups, semidefinite programs, and sums of squares*, 2002. In preparation.

[42] D. Gillespie, *The chemical langevin equation*, J. Chem. Phys, 113 (2000), pp. 297–306.

[43] S. Glavaski, A. Papachristodoulou, and K. Ariyur, *Safety verification of controlled advanced life support system using barrier certificates*, in Hybrid Systems: Computation and Control, 2005.

[44] D. Grigoriev and N. Vorobjov, *Complexity of Null- and Positivstellensatz proofs*, Annals of Pure and Applied Logic, 113 (2002), pp. 153–160.

[45] S. B. W. D. Group, *The systems biology workbench development group*, 2001.

[46] J. Hasty, J. Pradines, M. Dolnik, and J. J. Collins, *Noise-based switches and amplifiers for gene expression*, Proc. Natl. Acad. Sci. USA, 97 (2000), pp. 2075–2080.

[47] W. J. Hedley, M. R. Nelson, D. Bullivant, A. Cuellar, Y. Ge, M. Grehlinger, K. Jim, S. Lett, D. Nickerson, P. Nielsen, and H. Yu, *CellML specification*, tech. rep., 2001. Available on the Internet at http://www.cellml.org/public/specification/20010810/cellml_specification.html.

[48] W. J. Hedley, M. R. Nelson, D. P. Bullivant, and P. F. Nielson, *A short introduction to CellML*, Philosophical Transactions of the Royal Society of London A, 359 (2001), pp. 1073–1089.

[49] H. Hermjakob, *The HUPO PSI's molecular interaction format—a community standard for the representation of protein interaction data*, Nature Biotechnology, 22 (2004), pp. 177–183.

[50] M. Hucka, A. Finney, B. Bornstein, S. Keating, B. Shapiro, J. Matthews, B. Kovitz, M. Schilstra, A. Funahashi, J. Doyle, and H. Kitano, *Evolving a lingua franca and associated software infrastructure for computational systems biology: The Systems Biology Markup Language (SBML) project*, Systems Biology, 1 (2004), pp. 41–53.

[51] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novre, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang, *The Systems Biology Markup Language (SBML): A medium for representation and exchange of biochemical network models*, Bioinformatics, 19 (2003), pp. 524–531.

[52] Z. Jarvis-Wloszek, R. Feeley, W. Tan, K. Sun, and A. K. Packard, *Some controls applications of sum of squares programming*, in Proceedings of the IEEE Conference on Decision and Control, 2003.

[53] D. Jibetean and E. de Klerk, *Global optimization of rational functions: A semidefinite programming approach*. CWI Technical Report, 2003.

[54] H. K. Khalil, *Nonlinear Systems*, Prentice Hall, Inc., second ed., 1996.

[55] H. Kitano, *Cancer as a robust system: implications for anticancer therapy*, Nature Reviews Cancer, 4 (2004), pp. 227–235.

[56] B. Kovitz, *RFC 3823: MIME media type for the Systems Biology Markup Language (SBML)*, 2004. Available on the Internet at `http://www.faqs.org/rfcs/rfc3823.html`.

[57] H. Kurata, H. El-Samad, T. Yi, M. Khammash, and J. Doyle, *Feedback regulation of the heat shock response in* e. coli, in Proceedings of the 40th IEEE Conference on Decision and Control, 2001, pp. 837–842.

[58] H. J. Kushner, *Stochastic Stability and Control*, Academic Press, New York, 1967.

[59] A. D. Lander, *A calculus of purpose*, PLoS Biology, 2 (2004), pp. 0712–0714.

[60] J. B. Lasserre, *Global optimization with polynomials and the problem of moments*, SIAM J. Optim., 11 (2001), pp. 796–817.

[61] O. Lassila and R. Swick, *Resource description framework (RDF) model and syntax specification*, 1999. Available on the Internet at http://www.w3.org/TR/REC-rdf-syntax/.

[62] L. Li, D. Alderson, J. Doyle, and W. Willinger, *A first-principles approach to understanding the internet's router-level topology*, in ACM SIGCOMM, 2004.

[63] J. Lofberg and P. A. Parrilo, *From coefficients to samples: a new approach in SOS optimization*, in Proceedings of the 43rd IEEE Conference on Decision and Control, 2004.

[64] S. H. Low, F. Paganini, and J. C. Doyle, *Internet congestion control*, IEEE Control Systems Magazine, 22 (2002), pp. 28–43.

[65] S. H. Low, F. Paganini, J. Wang, S. A. Adlakha, and J. C. Doyle, *Dynamics of TCP/AQM and a scalable control*, in Proc. of IEEE Infocom, June 2002. `http://netlab.caltech.edu`.

[66] H. Mabuchi, M. Armen, B. Lev, M. Loncar, J. Vuckovic, H. J. Kimble, J. Preskill, M. Roukes, and A. Scherer, *Quantum networks based on cavity qed*, Quantum Information and Computation, 1 (2001), pp. 7–12.

[67] A. Megretski and A. Rantzer, *System analysis via integral quadratic constraints*, IEEE Transactions on Automatic Control, 42 (1997), pp. 819–830.

[68] P. Mendes, *Biochemistry by numbers: Simulation of biochemical pathways with Gepasi 3*, Trends in Biological Science, 22 (1997), pp. 361–363.

[69] P. Mendes, *Gepasi 3.21*. Available on the Internet at `http://www.gepasi.org`, 2001.

[70] M. Morita, Y. Tanaka, T. Kodama, Y. Kyogoku, H. Yanagi, and T. Yura, *Translational induction of heat shock transcription factor σ^{32}: Evidence for a built-in rna thermosensor*, Genes & Dev., 13 (1999), pp. 655–665.

[71] Y. Nesterov, *Squared functional systems and optimization problems*, in High Performance Optimization, J. Frenk, C. Roos, T. Terlaky, and S. Zhang, eds., Kluwer Academic Publishers, 2000, pp. 405–440.

[72] B. Øksendal, *Stochastic Differential Equation: An Introduction with Applications*, Springer-Verlag, Berlin, 2000.

[73] A. Packard, *Gain scheduling via linear fractional transformations*, Syst. Control Lett., 22 (1994), pp. 79–92.

[74] F. Paganini, J. C. Doyle, and S. H. Low, *Scalable laws for stable network congestion control*, in Proceedings of Conference on Decision and Control, December 2001. `http://www.ee.ucla.edu/ paganini`.

[75] F. Paganini, Z. Wang, S. H. Low, and J. C. Doyle, *A new tcp/aqm for stable operation in fast networks*, in Proceedings of IEEE Infocom, San Francisco, 2003.

[76] A. Papachristodoulou and S. Prajna, *On the construction Lyapunov functions using the sum of squares decomposition*, in Proceedings IEEE Conference on Decision and Control, 2002.

[77] ——, *Analysis of non-polynomial systems using the sum of squares decomposition*, in Positive Polynomials in Control, D. Henrion and A. Garulli, eds., Springer-Verlag, 2005, ch. 3, pp. 23–44.

[78] P. A. Parrilo, *Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization*, PhD thesis, California Institute of Technology, Pasadena, CA, 2000.

[79] ——, *Semidefinite programming relaxations for semialgebraic problems*, Mathematical Programming Series B, 96 (2003), pp. 293–320.

[80] P. A. Parrilo and R. Peretz, *An inequality for circle packings proved by semidefinite programming*, Discrete and Computational Geometry, 31 (2004).

[81] P. A. Parrilo and B. Sturmfels, *Minimizing polynomial functions*, in Workshop on Algorithmic and Quantitative Aspects of Real Algebraic Geometry in Mathematics and Computer Science, March, 1998.

[82] K. Poolla, P. Khargonekar, A. Tikku, J. Krause, and K. Nagpal, *A time-domain approach to model validation*, IEEE Transactions on Automatic Control, 39 (1994), pp. 951–959.

[83] S. Prajna, *Barrier certificates for nonlinear model validation*. To appear in Proceedings IEEE Conference on Decision and Control, 2003.

[84] ——, *Barrier certificates for nonlinear model validation*, Automatica, 42 (2006), pp. 117–126.

[85] S. Prajna and A. Jadbabaie, *Safety verification of hybrid systems using barrier certificates*, in Hybrid Systems: Computation and Control, LNCS 2993, Springer-Verlag, Heidelberg, 2004, pp. 477–492.

[86] ——, *Methods for safety verification of time-delay systems*, in Proceedings of the IEEE Conference on Decision and Control, 2005.

[87] S. Prajna, A. Jadbabaie, and G. J. Pappas, *Stochastic safety verification using barrier certificates*, in Proceedings of the IEEE Conference on Decision and Control, 2004.

[88] ——, *A framework for worst-case and stochastic safety verification using barrier certificates*. Submitted to *IEEE Transactions on Automatic Control*, 2005.

[89] S. Prajna and A. Papachristodoulou, *Analysis of switched and hybrid systems – beyond piecewise quadratic methods*, in Proceedings of the American Control Conference, 2003.

[90] S. Prajna, A. Papachristodoulou, and P. A. Parrilo, *Introducing SOSTOOLS: A general purpose sum of squares programming solver*, in Proceedings IEEE Conference on Decision and Control, 2002.

[91] ——, *SOSTOOLS – Sum of Squares Optimization Toolbox, User's Guide*. Available at `http://www.cds.caltech.edu/sostools` and `http://www.aut.ee.ethz.ch/~parrilo/sostools`, 2002.

[92] ——, *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*. Available from `http://www.cds.caltech.edu/sostools` and `http: //www.aut.ee.ethz.ch/~parrilo/sostools`, 2002.

[93] S. Prajna, P. A. Parrilo, and A. Rantzer, *Nonlinear control synthesis by convex optimization*. To appear in IEEE Transactions on Automatic Control, 2004.

[94] S. Prajna and A. Rantzer, *On the necessity of barrier certificates*, in Proceedings of the IFAC World Congress, 2005. To appear.

[95] ——, *Primal-dual tests for safety and reachability*, in Hybrid Systems: Computation and Control, Springer-Verlag, 2005. To appear.

[96] M. Ptashne, *A Genetic Switch: Phage λ and Higher Organisms*, Blackwell Scientific Publications and Cell Press, Cambridge, MA, 1992.

[97] B. Rahn, A. C. Doherty, and H. Mabuchi, *Exact and approximate analysis of concatenated quantum codes*, in Proc. of the Sixth International Conference on Quantum Communication, Measurement, and Computing, 2002.

[98] A. Rantzer and P. A. Parrilo, *On convexity in stabilization of nonlinear systems*, in Proceedings of the 39th IEEE Conf. on Decision and Control, vol. 3, 2000, pp. 2942–2945.

[99] C. Rao and A. Arkin, *Stochastic chemical kinetics and the quasi-steady-state assumption: Application to the gillespie algorithm*, J. Chem. Phys., 118 (2003), pp. 4999–5010.

[100] D. Reynolds, J. Carlson, and J. Doyle, *Design degrees of freedom and mechanisms for complexity*, Phys. Rev. E, 66 (2002), p. 016108.

[101] B. Reznick, *Extremal PSD forms with few terms*, Duke Mathematical Journal, 45 (1978), pp. 363–374.

[102] ——, *Some concrete aspects of Hilbert's 17th problem*, in Contemporary Mathematics, vol. 253, American Mathematical Society, 2000, pp. 251–272.

[103] M. Schweighofer, *Optimization of polynomials on compact semialgebraic sets*. Preprint, 2003.

[104] P. Seiler, *Stability region estimates for SDRE controlled systems using sum-of-squares optimization*, in Proceedings of the American Control Conference, 2003.

[105] J. Shamma and M. Athans, *Analysis of gain scheduled control for nonlinear plants*, IEEE Transactions on Automatic Control, 35 (1990), pp. 898–907.

[106] M. Shea and G. Ackers, *The or control system in bacteriophage lambda: A physical-chemical model for gene regulation*, J. Mol. Biol., (1985), pp. 211–230.

[107] N. Z. Shor, *Class of global minimum bounds of polynomial functions*, Cybernetics, 23 (1987), pp. 731–734.

[108] R. S. Smith and J. C. Doyle, *Model validation: a connection between robust control and identification*, IEEE Transactions on Automatic Control, 37 (1992), pp. 942–952.

[109] SourceForge.net, *The SourceForge open source software development web site*. Available on the Internet at `http://www.sourceforge.net/`, 2002.

[110] P. T. Spellman, M. Miller, J. Stewart, C. Troup, U. Sarkans, S. Chervitz, D. Bernhart, G. Sherlock, C. Ball, M. Lepage, M. Swiatek, W. L. Marks, J. Goncalves, S. Markel, D. Iordan, M. Shojatalab, A. Pizarro, J. White, R. Hubley, E. Deutsch, M. Senger, B. J. Aronow, A. Robinson, D. Bassett, C. J. S. Jr, and A. Brazma, *Design and implementation of microarray gene expression markup language (MAGE-ML)*, Genome Biology, 3 (2002), pp. 0046.1–0046.9.

[111] J. Stelling, U. Sauer, Z. Szallasi, F. Doyle, and J. Doyle, *Robustness of cellular functions*, Cell, 118 (2004), pp. 675–685.

[112] G. Stengle, *A Nullstellensatz and a Positivstellensatz in semialgebraic geometry*, Mathematische Annalen, 207 (1974), pp. 87–97.

[113] D. Straus, W. Walter, and C. Gross, *Dnak, dnaj, and grpe heat shock proteins negatively regulate heat shock gene expression by controlling the synthesis and stability of σ^{32}*, Genes Dev., 4 (1990), pp. 2202–2209.

[114] L. Strömbäck and P. Lambrix, *Representations of molecular pathways: An evaluation of SBML, PSI MI and BioPAX*, Bioinformatics, 21 (2005), pp. 4401–4407.

[115] J. F. Sturm, *Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones*, Optimization Methods and Software, 11–12 (1999), pp. 625–653. Available at `http://fewcal.kub.nl/sturm/software/sedumi.html`.

[116] B. Sturmfels, *Solving Systems of Polynomial Equations*, American Mathematical Society, 2002.

[117] M. Sznaier, A. C. Doherty, M. Barahona, H. Mabuchi, and J. C. Doyle, *A new bound of the $l_2[0,t]$-induced norm and applications to model reduction*, in Proc. of the American Control Conference, 2002.

[118] R. Tanaka and J. Doyle, *Scale-rich metabolic networks: background and introduction*, tech. rep., arXiv:q-bio.MN/0410009, 2004.

[119] A. van der Schaft and H. Schumacher, *An Introduction to Hybrid Dynamical Systems*, Springer-Verlag, London, 2000.

[120] L. Vandenberghe and S. Boyd, *Semidefinite programming*, SIAM Review, 38 (1996), pp. 49–95.

[121] J. Wang, L. Li, S. H. Low, and J. C. Doyle, *Can tcp and shortest path routing maximize utility?*, in Proceedings of IEEE Infocom, San Francisco, 2003.

[122] H. Wolkowicz, R. Saigal, and L. Vandenberghe, eds., *Handbook of Semidefinite Programming*, Kluwer, 2000.

[123] V. A. Yakubovic, *S-procedure in nonlinear control theory*, Vestnik Leningrad University, 4 (1977), pp. 73–93. English translation; original Russian publication in *Vestnik Leningradskogo Universiteta, Seriya Matematika*, Leningrad, Russia, 1971, pp. 62–77.

[124] Y. Yamada and J. Primbs, *Distribution based option pricing on lattice asset dynamics models*, in Proc of the 2001 American Control Conference, 2000, pp. 3393–3397.

[125] ——, *Construction of multinomial lattices for optimal hedges*, in Proc. of the International Conference on Computational Science, 2001.

[126] ——, *Construction of multinomial lattices for optimal hedges*, in Lecture Note Series on Computer Science, Springer Verlag, 2001, pp. 579–588.

[127] ——, *Risk estimates for dynamic hedging using convex probability bounds*, in Proc of the 2001 American Control Conference, 2001.

[128] ——, *Distribution based option pricing on lattice asset dynamics models*, International Journal of Theoretical and Applied Finance, 5 (2002), pp. 599–618.

[129] T.-M. Yi, M. Fazel, X. Liu, T. Otitoju, J. Goncalves, A. Papachristodoulou, S. Prajna, and J. Doyle, *Application of robust model validation using SOSTOOLS to the study of G-protein signaling in yeast*, in Proc. Foundations of Systems Biology and Engineering, August 2005.

[130] T.-M. Yi, Y. Huang, M. Simon, and J. C. Doyle, *Robust perfect adaptation in bacterial chemotaxis through integral feedback control*, in Proceedings of the National Academy of Sciences, USA, vol. 97, 2000, pp. 4649–4653.

[131] T.-M. Yi, H. Kitano, and M. I. Simon, *A quantitative characterization of the yeast heterotrimeric g protein cycle*, Proc. Natl Acad Sci U.S.A., 100, pp. 10764–9.

[132] T. Zhou, J. Carlson, and J. Doyle, *Mutation, specialization, and hypersensitivity in highly optimized tolerance*, Proc. Nat. Acad. Sci., 99 (2002), pp. 2049–2055.

[133] X. Zhu, J. Yu, and J. C. Doyle, *Heavy tails, generalized coding, and optimal web layout*, in Proceedings of IEEE Infocom, April 2001.